Appendix A from A. Moilanen, “Reserve Selection Using Nonlinear Species Distribution Models”  
(Am. Nat., vol. 165, no. 6, p. 695)

The Stochastic Spatial Search Algorithm

The search over large-scale spatial patterns was done using a genetic algorithm (GA; Holland 1975; Goldberg 1989; Jaramillo et al. 2002). The technique uses principles of evolution to solve optimization problems. A GA belongs to the class of metaheuristics, which are general, usually stochastic, search strategies that are not problem specific (see Blum and Roli 2003 for review). In the GA description, I concentrate on the spatial crossover component, which is the most important feature of the algorithm. For references and a more comprehensive discussion about GAs in the context of spatial optimization in ecology, see an article by Moilanen and Cabeza (2002).

The GA uses a population of $GA_M$ potential solutions in the search process, $x^{(1)}$, $x^{(2)}$, …, $x^{(M)}$. Using GA terminology, each solution has an associated fitness value, $F(x)$, calculated from equation (2). In this context, high fitness means a low value for the objective (solution size), which is minimized. At each generation of the GA, only a given proportion $GA_R$ (0.15 was used) of the population with the highest associated fitness values (smallest solution sizes) reproduces; this selection guides the search process toward solutions with good (small) values for the objective function. The GA is run for $GA_G$ generations. The population (the interim solutions) for generation $g$ is produced from the population of generation $g - 1$ by randomly picking pairs of parents from the $GA_R \times GA_M$ solutions having the highest fitness values. Combining the parent solutions in an operation called crossover, or recombination, produces two children with properties from both parents. In practical terms, spatial crossover here means a swap of a couple of rectangular blocks of land between two candidate solutions. The children are inserted into the population of generation $g + 1$. Recombination is a critical component of GAs, separating them from other metaheuristic search methods. Iteration of the GA with repeated selection of the fittest and recombination causes the search process to converge to promising regions of the search space. Another component of the GA is mutation, which is an addition of a new block of area to the selection or the deletion of a block of selected area. Crossover and mutation are search operators that maintain a diversity of different solutions and produce new solutions for evaluation.

The GA used here includes nonstandard components in the implementation of recombination and mutation. The typical crossover operator used by GAs takes the two parent solutions (coded into binary character strings) and swaps elements (bits; here grid cells) between them. In the so-called uniform crossover, each bit is swapped between parents with a given probability (often 0.5) to produce a mixing of the parents. The standard binary crossover operators are ill suited for the present problem because they are too disruptive, ignoring the need for spatial structure and cohesiveness in the children. For example, consider two solutions (large grids) where solution $X = \{A, B\}$ has regions A and B selected and solution $Y = \{C, D\}$. Assume that the best solution is actually $\{B, C\}$. How do you get to $\{B, C\}$? The typical binary crossover operators will not be able to generate such a solution, but rather the children will include a complete mix of all regions A, B, C, and D, which will be no good especially if connectivity and spatial continuity of reserves is important. Consequently, the crossover was implemented in the present algorithm as a swap of randomly placed rectangular multicell blocks of land between the parents. Thus, major changes can be made to solutions while retaining most of the detailed spatial structure present in the parents.

The number of blocks swapped between parents to produce the children is given by parameter $GA_{X_N}$ and the sizes of the blocks swapped by $GA_{X_S}$. The size of a block in the $x$-dimension is given by $u \times GA_{X_S} \times grid_{x-dim}$, where $u$ is a uniform (0, 1) deviate and $grid_{x-dim}$ is the dimension of the landscape in the $x$-dimension. The size of the crossover block in the $y$-dimension is randomized similarly.

Mutation is a GA component that maintains variation in the population. Here it was implemented as an addition of blocks to or a removal of blocks from the solution. One mutation per solution happens with
probability $GA_{MP}$ to a randomly selected location in the solution grid. The size of the mutation block is randomized in the same way as the size of a crossover block, using parameter $GA_{MS}$.

Because different landscapes might require optimal solutions with different spatial characteristics, it is impossible to tell which values of $GA_{XN}$, $GA_{XS}$, $GA_{MP}$, and $GA_{MS}$ would be best suited for optimizing a particular problem. Therefore, these parameters were made self-adaptive (Beyer and Deb 2000). Each GA individual carries a copy of each of these four parameters. The parameters are crossed in recombination by randomizing them to the interval defined by the parents’ values. The control parameters were also mutated by rate 0.1, where mutation multiplied the value of the variable by a normal deviate having mean 1 and SD 0.1. Consequently, the selection and recombination also adapts the parameters of the recombination and mutation to the characteristics of the problem. Values for control parameters for individuals in the initial population of the GA were randomized from the intervals $GA_{XN} \in (1, 10)$, $GA_{XS} \in (0, 0.3)$, $GA_{MP} \in (0, 1)$, and $GA_{MS} \in (0.05, 0.5)$, and subsequently, the parameter values were allowed to evolve freely in the GA.

**Local Improvement of Solutions**

The reserve structures generated by the GA are, in general, suboptimal. To start with, either the solution will usually fail to meet the target for some species or, alternatively, there might be too much area in the solution, and some cells could be removed, thus improving the efficiency (fitness) of the solution without violating the target requirements. The idea of the local improvement is to add to (if below targets) or remove cells from the solution (if above targets) until the solution is at the target or just barely above it for all species. (Note that unlike crossover, local search operates on individual cells, not blocks of land.) If the local search succeeds in improving the solution, the changes are carried over back to the GA. The following algorithm implements the local search.

**ALGORITHM evaluate and improve solution**

1. Let $f_i = F(x)$ from equation (2). This step requires iterating equations (4) and (5) until convergence.
2. **REPEAT LS\(_{\text{rnds}}\) (\(=3\)) times**
   2.1 Let $\tau = [R_i(x) - T_i]/T_i$. Let $r_{\text{min}} = \min_j r_j$
   2.2 IF ($r_{\text{min}} < 0$), iteratively add cells to solution generating solution $x'$ updating $R_i(x')$
   ELSE
   iteratively remove cells from $x$ generating $x'$ until no cell can be removed without $\min_i [R_i(x') - T_i]$ becoming < 0.
   2.3 Let $f_2 = F(x')$. IF ($f_2 < f_1$), replace $x$ with $x'$ and let $f_1 = f_2$.
3. **RETURN possibly modified $x$ and the respective $F(x)$ to the GA.**

The main question left open in the algorithm above is the identity of cells added/removed at step 2.2, which determines whether the above procedure is effective. Cell addition/removal was based on the quantities

$$\delta_i = \sum_j \frac{p_{x_{ij, \text{max}}}[0, T_i - R_i(x)]}{c_i[R_i(1) - R_i(x)]}$$

when adding or

$$\delta_i = \sum_j \frac{p_{x_{ij, \text{min}}}}{c_i[R_i(x) - T_i][R_i(1) - R_i(x)]} \quad (A1)$$

when removing.

For example, when adding sites not already included, it is good to have high occurrence (high $p_{x_{ij, \text{max}}}$) for species with large unmet targets $[T_i - R_i(x)]$, few unselected populations remaining (small $R_i(1) - R_i(x)$), and low cost. ($R_i(1)$ is representation for species when the full landscape is selected, $x_i = 1$ for all $i$) A high value for $\delta_i$ will thus indicate a good site to add to the solution. Similar reasoning applies for site removal, except the first candidate for removal has the lowest $\delta_i$ and only sites with $x_i = 1$ are considered. For simplicity, the following is written from the perspective of adding sites, but a fully analogous line of argument applies for cell removal.

Essentially, the local search is about finding a computationally inexpensive local improvement method that has
App. A from A. Moilanen, “Nonlinear Reserve Selection”

A reasonable success rate in improving the solution. A highly reliable local search process is not very useful if using it requires unreasonable computation, as would be the case in the present context if species distribution models were to be iterated until convergence following each minute change in the landscape structure during the search process. Ideally, local search would add one site at a time to the solution using equation (A1) to rank cells and to find the optimal one to add. Then, the site would be added to $x$, $p_{ij}$, and $R_j(x)$ would be updated, $\delta_i$ would be updated for all sites, and the cycle would begin anew. This procedure would be analogous to a gradient search for continuous optimization. Unfortunately, this is too expensive computationally, $O(N^2 \log N)$, to work satisfactorily in landscapes where there are tens of thousands of cells that need to be considered for addition and potentially thousands of cells to be added during one local search round. Therefore, equation (A1) was implemented in an approximate manner by compiling a long list of potential changes to the reserve and doing many in one pass.

Specifically, $\delta_i$ was calculated once for all sites, sites were ranked by $\delta_i$, and then sites were added until targets were met. Importantly, this was done without recomputation of $p_{ij}$ and $\delta_i$ and without reranking of available cells following each landscape change. In essence, local search was done analogously to a scoring method for reserve selection (Smith and Theberge 1986), which is suboptimal when many sites are added simultaneously but has the advantage of running in $O(N \log N)$ time instead of $O(N^2 \log N)$ time. (With the scoring approach, the vector is sorted only once.)

If the local search were the only thing used for optimizing the solution, the scoring-type approach would be insufficient because the nonlinear feedback loop between the landscape structure and probabilities of occurrence would be ignored. But in combination with the GA, occasional failures of the local search procedure are of little significance because they do not affect the rest of the search process. And, the local search is close to optimal when the solution coming from the GA is close to the target levels, and thus only a few changes need to be made to it.

Two different ways (chosen by a probability .5) of selecting sites based on $\delta_i$ were used. For a local search round (item 2, above), sites were either added from the top of the list ranked by $\delta_i$ or added by picking them in random order with probability directly proportional to $\delta_i/\sum \delta_i$. These variants are included to maintain variation in the search process. The former one improves solutions faster, but it has the weakness of always trying the same improvements when starting from the same initial structure, which is unsatisfactory for a multimodal optimization problem.

Because the search process is stochastic and the local search is approximate, there is no guarantee that a solution improved after the GA by three rounds of local search using the algorithm above actually meets targets for all species. This is a problem because if the optimal solution is located on the boundary between feasible and infeasible, removing just one site from the solution will cause the target to be failed for at least one species. Thus, a close to optimal solution might be just barely infeasible, but such solutions should not be removed from the search process as total failures. To allow such solutions to continue in the GA, a penalty function was used; $F$ was replaced by $F' = F(1 + d_{\min})^5$ if the solution was below the target ($d_{\min} < 0$). This results in a $\sim 30\%$ ($70\%$) increase in the value of the objective function for a solution 5% (10%) below target, which indicates that only solutions slightly below the target are able to persist in the GA population. Naturally, an infeasible solution was not allowed to become the optimal solution for the problem.

The optimization runs used GA population size 300 and 30 GA generations. Given timings are for a 3-GHz Pentium 4 workstation. Software implementing the algorithm is downloadable from the metapopulation research group Web site http://www.helsinki.fi/science/metapop. The implementation includes the ability to automatically include already reserved areas into the selection process or to exclude others from it. An option for using the boundary length penalty (Possingham et al. 2000; Nalle et al. 2002; Fischer and Church 2003; Cabeza et al. 2004a) for adding additional aggregation to the solutions is included.