



# Mathematical methods for spatially cohesive reserve design

Mark D. McDonnell<sup>a</sup>, Hugh P. Possingham<sup>b</sup>, Ian R. Ball<sup>c</sup> and Elizabeth A. Cousins<sup>a</sup>

<sup>a</sup> Department of Applied Mathematics, The University of Adelaide, South Australia 5005, Australia

<sup>b</sup> Department of Mathematics and Zoology & Entomology, The University of Queensland, St Lucia Queensland 4072, Australia

<sup>c</sup> Australian Antarctic Division, Channel Highway, Kingston 7050, Tasmania, Australia

The problem of designing spatially cohesive nature reserve systems that meet biodiversity objectives is formulated as a nonlinear integer programming problem. The multiobjective function minimises a combination of boundary length, area and failed representation of the biological attributes we are trying to conserve. The task is to reserve a subset of sites that best meet this objective. We use data on the distribution of habitats in the Northern Territory, Australia, to show how simulated annealing and a greedy heuristic algorithm can be used to generate good solutions to such large reserve design problems, and to compare the effectiveness of these methods.

**Keywords:** reserve design, simulated annealing, set covering problem, spatial, clustering, fragmentation, optimisation heuristics, multiobjective optimisation

## 1. Introduction

Many countries have committed to conserving significant amounts of their native biodiversity [1]. Biodiversity includes the diversity of ecosystems and the diversity between and within species. Establishing reserve systems dedicated to the conservation of biodiversity is the cornerstone of most national, regional and state conservation strategies [2]. In this paper, we explore the question of which parcels of land, henceforth termed sites, should be selected for reservation. From the perspective of nature conservation alone, one would attempt to have the largest reserve system possible; however, in reality, the extent of any reserve system will be limited by social and economic constraints. Thus, building a reserve network that will conserve biodiversity effectively is not a process of accumulating as much land as possible, but necessitates choosing sites as efficiently as possible from those available [3]. Here, “efficiently” means minimising the “cost” of the reserve system that meets certain biodiversity goals.

We focus on a particular class of reserve design problems where the goal is to achieve some minimum representation of biodiversity features for the smallest possible cost. In such problems the objective is to minimise costs, and biodiversity enters as a constraint. One possible biodiversity constraint is to ensure that every type of vegetation that exists within a region is represented a given number of times (that is, appears in a given number of separate sites) in a reserve system. Ideally we would like to minimise the total monetary cost required for purchasing land that meets this constraint. However, since this information is often not readily available, the total area of selected sites (or number of sites, if all sites have equal area) can act as a surrogate for the monetary cost. If the chosen goal is to select the minimum number of sites, or minimum total area of reserved sites that meets the above constraint, we have a minimum set-covering problem. This can be formulated as an inte-

ger linear programming problem, or ILP [4]. Similarly, we might require a certain percentage of the total area of each species to be represented in the final reserve system. Each of these problems may be too large to solve using mathematical programming methods that guarantee finding an optimal solution, such as the branch and bound method [5]. However a number of heuristics (approximate methods), including rarity and greedy based algorithms [6], and a random search technique known as *simulated annealing* [7], have been applied to real data sets to obtain a range of good solutions to these problems. It should be noted that whilst it is desirable to find optimal or near optimal solutions to our objectives, it is often more important to be able to provide a range of good solutions from which to choose [8]. Heuristic methods more readily allow us to obtain a range of solutions than techniques such as the branch and bound method.

## 2. Spatially cohesive reserve design

One limitation of the minimum set-covering problem is that it does not account for the spatial relationships between the sites selected for the reserve system. Without some modification or additional constraints, the final reserve system will almost always be highly fragmented, that is, a largely disconnected set of sites. This may be a major problem as there are both ecological and economic reasons why reserves should be spatially contiguous with low edge to area ratios. Disconnected reserve systems, as well as long thin reserves, tend to have high edge to area ratios and will be more vulnerable to weed and pest invasions, as well as to other *edge effects*, caused by biotic interactions like predation [9] or abiotic factors like humidity or wind. From an economic perspective, the cost of management often scales more closely with the boundary length of a reserve than with reserve area. For example, boundaries may need to be maintained or patrolled, and longer boundaries usually mean

more neighbours. Hence we would generally like to cluster sites in a reserve system in order to minimise the edge to area ratio. However, there may also be compelling reasons to avoid clustering. Where catastrophes can impact large areas and cause local extinctions, it may be better to reserve each species in two or more separate places, thereby reducing the risk posed by a single contiguous reserve.

When clustering is desired, the clustering of reserved sites is achieved in some algorithms by including an adjacency constraint where, all else being equal, sites are added if they are next to sites already chosen [10]. Another way to promote clustering, incorporated here, is to minimise the boundary length of the reserve system. For a given area, a smaller boundary length gives a more compact shape. Thus, overall reserve boundary length becomes an important factor and the reserve design problem becomes more complex. There are now two objectives that must be traded off against each other: minimising boundary length and minimising the total area of reserved sites. We formulate this problem as a bi-objective optimization problem. Our formulation is nonlinear, because the “cost” of adding a site (change in the value of the objective function – see section 3) to the reserve system depends on which other sites are already reserved, and on the spatial relationships between candidate sites and those already reserved.

In order to minimise a suitable combination of both boundary length and area in this problem, we introduce an *objective function weight* that we will call boundary length modifier (BLM) [11]. BLM will be used to weight the cost of the boundary length compared to the cost of area alone. The technique of combining two or more objectives into a single objective function is known as the *weighting method* of multiobjective programming [12].

The objective to be minimised is now the sum of reserve area and BLM times the reserve boundary length. By varying BLM, the relative importance of compactness and size can be changed. If BLM is set to zero, then boundary length is ignored. The objectives addressed in this paper are similar to objectives considered previously in land acquisition or land allocation problems, where objectives of minimizing area and minimizing boundary length are traded off against each other [13,14].

In this paper we compare the results obtained by applying a greedy heuristic algorithm and simulated annealing to the problem of designing reserve systems where varying importance or weight is placed on boundary length. We illustrate our method using data on the distribution of vegetation types in the Northern Territory, Australia. Before proceeding to describe the results obtained from this example, we give a mathematical formulation of the problem, and briefly describe the two solution methods used.

### 3. Mathematical formulation

Recall that the objective is to design a reserve system by selecting from all available sites a subset of sites that would

minimise the “cost” (in this case taken to be a linear combination of area and boundary length of the reserve system), whilst ensuring that all conservation values are sufficiently represented. For example, if the conservation values to reserve are types of vegetation, one criterion for representation is that the selected sites contain a specified minimum percentage of the total area occupied by each type. Here, we use the term *conservation value* to refer to entities we wish to include in the reserve system. These could include specific flora or fauna species as well as types of vegetation, ecosystems or landforms. Here, we restrict the discussion to the goal of designing a reserve system that contains a certain *area* of each conservation value.

To describe the problem mathematically for a given data set, let

$m$  = total number of sites available for inclusion in the reserve system;

$n$  = total number of conservation values present in these  $m$  sites;

$d_i$  = total area of site  $i$  (in  $\text{km}^2$ ),  $\forall i = 1, 2, \dots, m$  ( $d_i > 0$ );

$a_{ik}$  = area (in  $\text{km}^2$ ) of conservation value  $k$  on site  $i$ ,  $\forall i = 1, 2, \dots, m, \forall k = 1, 2, \dots, n$  ( $a_{ik} \geq 0$ );

$b_{ij}$  = length of shared boundary (in km) between sites  $i$  and  $j$ ,  $\forall i = 1, 2, \dots, m, \forall j = 1, 2, \dots, m$  ( $b_{ij} \geq 0$  and  $i \neq j$ );

$\beta_i$  = total boundary length (in km) of site  $i$ ,  $\forall i = 1, 2, \dots, m$  ( $\beta_i > 0$ ).

Also, let

$c_k$  = required area (in  $\text{km}^2$ ) of conservation value  $k$ ,  $\forall k = 1, 2, \dots, n$  ( $c_k \geq 0$ );

BLM = Boundary Length Modifier

( $c_k$  and BLM are adjustable parameters specified prior to running the algorithms).

A state vector  $\mathbf{x}$  of control variables indicates whether sites are included in the reserve:

$$\mathbf{x} = (x_1, x_2, \dots, x_m), \quad \text{where}$$

$$x_i = \begin{cases} 1, & \text{if site } i \text{ is included in the reserve,} \\ 0, & \text{otherwise.} \end{cases}$$

So if  $\mathbf{X}$  is the set of all possible state vectors  $\mathbf{x}$ , then mathematically we wish to find the vector  $\hat{\mathbf{x}} \in \mathbf{X}$  which minimises the “cost” of the reserve, while meeting representation constraints. Our *objective* or *cost function* here is the sum of area and weighted boundary length:

$$C(\mathbf{x}) = \text{BLM} \times B(\mathbf{x}) + \sum_{i=1}^m d_i x_i, \quad (1)$$

where the total boundary length of the reserve  $B(\mathbf{x})$  is given by

$$B(\mathbf{x}) = \sum_{i=1}^m \beta_i x_i - 2 \sum_{i=1}^{m-1} \sum_{j=i+1}^m b_{ij} x_i x_j. \quad (2)$$

Thus, the problem of minimising the cost,  $C(\mathbf{x})$ , subject to the representation constraints can be written as the integer quadratic program (IQP):

$$\begin{aligned} \text{Minimize } C(\mathbf{x}) &= \text{BLM} \times B(\mathbf{x}) + \sum_{i=1}^m d_i x_i \\ \text{subject to: } \sum_{i=1}^m a_{ik} x_i &\geq c_k, \quad \forall k = 1, 2, \dots, n, \\ x_i &\in \{0, 1\}, \quad \forall i = 1, 2, \dots, m. \end{aligned} \quad (\text{IQP})$$

Note that the above formulation is a generalisation of other reserve design problems. If our interest is in minimising the total area of a reserve system, with no consideration of boundary length, then BLM is set to zero in (IQP). Further simplifications can be made, with the simplest being the problem of representing all conservation values at least once, in the fewest possible sites [4,7]. The effectiveness of various solution methods for these problems is discussed in [6,7].

#### 4. Solution methods

Many 0–1 integer programs are known to be NP-complete (e.g., the set covering problem) [15], and it is highly likely that (IQP) is also NP-complete. This means that the difficulty of finding an exact optimal solution increases exponentially with the number of constraints. Hence, heuristic methods are often used to obtain good, and hopefully near optimal, solutions to NP-complete problems.

Many heuristics require the minimisation of an *unconstrained* cost function. To allow the application of such heuristics to problems like (IQP), constraints are incorporated into a cost function using weighted penalty factors. In the current problem, we use a separate penalty factor,  $f_k$ , for each conservation value. These penalty factors replace each of the representation constraints in (IQP) by adding to the cost function  $C(\mathbf{x})$  for each unmet constraint. The formulation given in (IQP) can now be replaced by problem (IQP\*):

$$\text{Minimize } C(\mathbf{x}) = \text{BLM} \times B(\mathbf{x}) + \sum_{i=1}^m d_i x_i + \sum_{k=1}^n f_k. \quad (\text{IQP}^*)$$

The principle we use for calculating each penalty,  $f_k$ , is that if the target area for a conservation value has not been met, then the penalty should be the cost required to raise the reserved area of that conservation value up to its target area [11].

Ideally, we would recalculate  $f_k$  to be equal this cost after every change made to a reserve system within a solution method. However, this would be very time consuming in an algorithm where a very large number of changes are made, for example the simulated annealing method described below. Instead, we calculate the minimum cost,  $w_k$ , (as given by equation (1)) required to meet the target area for conservation value  $k$  alone. After each change to a reserve system, we set the current penalty for each conservation value to be

$w_k$  multiplied by the fraction of the target area that has not been met.

To express this mathematically, we need to introduce the following *slack* variable,  $u_k$ :

$$u_k = \begin{cases} c_k - \sum_{i=1}^m a_{ik} x_i, & \text{if } \sum_{i=1}^m a_{ik} x_i < c_k, \\ \forall k = 1, 2, \dots, n; \\ 0, & \text{otherwise.} \end{cases}$$

Thus,  $u_k$  is the target area of conservation value  $k$  less the area of conservation value  $k$  that is included in the current reserve system, *unless* the target area for conservation value  $k$  has been met, in which case  $u_k$  is zero. So  $u_k$  is the shortfall in area between what is needed and what has been reserved.

Then for  $c_k > 0$  the penalty for conservation value  $k$  is given by equation (3). If  $c_k$  is zero for some  $k$  then  $f_k$  is zero.

$$f_k = \frac{w_k u_k}{c_k}, \quad \forall k = 1, 2, \dots, n. \quad (3)$$

Initially, when no sites are reserved, the penalty for each conservation value is at its maximum value,  $w_k$ . As sites are reserved, the slack in conservation value  $k$ ,  $u_k$ , decreases, and the penalty for that conservation value also decreases until the target has been reached and the penalty becomes zero.

We estimate  $w_k$  using the greedy heuristic described below, except that each conservation value is considered in isolation from the others.

There are several heuristics that can be applied to a problem of this kind. We will concern ourselves here with a *greedy algorithm* and *simulated annealing*. These heuristics have an element of randomness, and therefore can be run a large number of times to obtain a wide range of different solutions. A brief description of each is given in the following two sections. Note that heuristic methods that incorporate a preference for selecting sites that contain rare conservation values may produce better results than a greedy heuristic [6]. However, we have found that greedy heuristics have an advantage in terms of ease of implementation, both in coding of the algorithm into software and in speed of execution, especially when spatial relationships are included.

##### 4.1. The greedy algorithm

Greedy algorithms work in general as their name suggests: an *existing solution* to a problem is updated to what is called a *neighbouring solution* (in the solution space), chosen so as to give the biggest decrease (for a minimisation problem) in the present value of the objective function. This repeats until no further decrease in the objective function is possible. In our application of a greedy algorithm to problem (IQP\*), the initial existing solution consists of all sites unreserved. At each subsequent iteration, the existing solution is a selection of sites presently included in the design. A neighbouring solution is then a reserve system where a single site

has been added to the design. Sites are only added to the reserve system, and never removed, and thus this implementation is known as a *greedy adding algorithm*. Since the objective is to minimise  $C(\mathbf{x})$ , the site that is added to the design at each iteration is the one that gives a neighbouring solution that provides the largest decrease in the value of  $C(\mathbf{x})$ . This neighbouring solution then becomes the existing solution in the next iteration. When two or more neighbouring solutions tie for the largest decrease in  $C(\mathbf{x})$ , the neighbouring solution selected is chosen randomly. Note that when all sites are unreserved  $x_i = 0 \forall i$  and  $u_k = c_k \forall k$ , and thus the penalty term commences at its maximum value and can only decrease as sites are reserved. The greedy algorithm stops when  $u_k = 0 \forall k$ .

For the example considered in section 5 of this paper, repeated application of this algorithm gave very few different solutions, since only rarely did ties occur for the largest decrease in  $C(\mathbf{x})$ . To obtain a larger variety of alternative solutions for this problem we used a modified version the algorithm, where a tie was considered to have occurred whenever two or more sites give decreases in  $C(\mathbf{x})$  within a set small percentage of the site with largest decrease in  $C(\mathbf{x})$ .

The greedy algorithm described above will possibly result in some redundant sites being included. That is, the final solution given by the greedy algorithm may include some sites that have increased  $C(\mathbf{x})$ , without adding to the biodiversity aim. These can be removed by applying an *iterative improvement* algorithm in conjunction with the greedy algorithm. This algorithm removes any reserved sites that cause  $C(\mathbf{x})$  to decrease without  $u_k$  becoming nonzero for any conservation value  $k$ .

In general, greedy algorithms are quite fast, and provide good solutions to combinatorial optimisation problems. Their main drawback is that they might converge to a local optimum, and might never find a global optimal solution. To overcome the problem of converging to a local optimum (local minimum in this case), it is necessary to provide a means for sometimes choosing a neighbouring solution that will temporarily increase the value of the objective function,  $C(\mathbf{x})$ , to force solutions away from poor local minima, and hopefully towards better local minima, or a global minimum. One such method is simulated annealing.

#### 4.2. Simulated annealing

*Simulated annealing* is an optimisation method that is based on the roughly analogous physical process of heating and then slowly cooling a substance (for example glass or steel) to obtain a strong crystalline structure [16,17]. The physical process follows a schedule of time periods and temperatures, called an *annealing schedule*. An annealing schedule of “temperatures”, and “times” for changing the temperature, is also specified in *simulated annealing*. Here, the “temperature” at a certain “time” is a parameter for a function specifying whether a certain neighbouring solution to a current solution in an optimisation problem should be

accepted as a new current solution or not. This function is called the *acceptance function*.

The simulated annealing algorithm performs many iterations, the number of which is specified in advance by the user. When simulated annealing is applied to (IQP\*), at each iteration the existing solution (a selection of sites presently included in the design) is modified to give a neighbouring solution by randomly selecting a site and either reserving it, if not already reserved, or unreserving it otherwise. This modification is then either accepted or rejected, according to the acceptance function, and the subsequent solution is then used in the next iteration. The acceptance function is dependent on the current temperature in the annealing schedule of the algorithm. Let

$T$  = the current temperature in an annealing schedule, with  $T > 0$ ,  
 $\Delta C$  = the change in objective function given by modifying an existing solution  $\mathbf{x}'$  to a neighbouring solution  $\mathbf{x}''$ .  
 Hence,  $\Delta C = C(\mathbf{x}') - C(\mathbf{x}'')$ .

Note that a decrease in cost function means that  $\Delta C$  is positive, and an increase means  $\Delta C$  is negative. The acceptance function is a function giving the probability,  $P$ , of accepting the modification of the current solution:

$$P(\Delta C, T) = \text{minimum}(1, e^{\Delta C/T}). \quad (4)$$

Thus a neighbouring solution that results in decreasing costs is always accepted, since  $\Delta C$  is positive and therefore  $P = 1$ . A neighbouring solution that results in increasing costs (so that  $\Delta C$  is negative) is accepted with probability  $e^{\Delta C/T}$ . Therefore, an annealing schedule that specifies values of  $T$  that decrease as the computations progress ensures the probability of accepting the “bad” modification decreases as time progresses. This occasional acceptance of “bad” changes allows convergence to local minima to be avoided early on in the algorithm, and increases the probability of obtaining an optimal or near optimal solution. Selection of a good annealing schedule is very important to the success of a simulated annealing scheme, and various rules of thumb for designing annealing schedules exist in the literature [17,18]. A key factor in the success of an annealing schedule is the number of times,  $N$ , that a neighbouring solution is considered. In general, the higher the value of  $N$ , the better the final solution. However, increasing  $N$  also means increasing the computer run time required for running the algorithm.

It is known that under certain conditions, simulated annealing algorithms will asymptotically converge to the optimal solution given infinite time [17,18]. In practice, only finite time is available, and so usually only a good feasible solution is obtained. To increase the probability of finding better solutions, the algorithm can be applied a large number of times.

## 5. Examples

We now compare the performance of the greedy algorithm with simulated annealing when applied to problem

(IQP\*) by examining a specific example. For the same set of data, we obtain solutions for two different representation targets and a range of values for the boundary length modifier for each target (including the special case of BLM equal to zero). The data set used contains the areas of 112 vegetation types within a grid of 1958 equally sized rectangles in the Northern Territory, Australia. For this example, the conservation values are vegetation types, and the sites are areas of land (with each side 0.25 degrees in length). The grid is 36 sites wide (from west to east) by 60 sites high (from south to north), with only land based sites being included. We assumed that boundaries on the edges of the grid and boundaries adjacent to sea are no more or less costly than boundaries within the grid. We also assume for simplicity that all the sites are square with equal area, and have sides with equal length in km. Note that the algorithms we use in this paper are also suitable for data where sites have irregular boundaries and non-equal areas [3].

The two different conservation targets considered are 5 and 10% area. That is, the goal is to reserve sufficient sites such that all 112 conservation values have 5% (or 10%) of their total area included in the reserved sites, whilst minimising the weighted sum of area and boundary length.

### 5.1. Zero BLM

Here, boundary length is disregarded completely (BLM = 0), and only the minimisation of the total area of the reserve system is considered. Since each site has equal area, this problem is equivalent to minimising the total number of sites needed to meet the conservation target and is thus the set-covering problem considered in [4].

A simple change was made to the basic greedy algorithm given in algorithm 1, which greatly improved its performance. The change was to add a *bonus factor* to the change in  $C(\mathbf{x})$  for site  $i$ , if for any conservation value  $k$  on that site, the area of the conservation value was enough to make the slack in that conservation value zero, that is,  $a_{ik} > u_k$ . The improved algorithm will be referred to as *Improved Greedy*. This improvement method is not used for nonzero values of BLM, as it would result in area becoming more important than the value of BLM dictates it should, and the value of BLM would lose its meaning.

Table 1 summarizes the results obtained for 10 runs of each of the basic greedy algorithm, the improved greedy algorithm and simulated annealing. The table gives the minimum, maximum and median values (over the 10 runs) of the number of sites reserved by each algorithm. It also gives the computer run time required for the 10 runs of each algorithm, for software written in C++ and run on a Pentium III processor. From table 1, it is clear that the improved greedy algorithm has given the best results for this problem – better results than simulated annealing, and in a much shorter time.

By varying the number of iterations in a simulated annealing run,  $N$ , the run time can be substantially increased or decreased, with a resulting decrease or increase in the median value. To illustrate this, table 2 shows the results obtained

Table 1  
Results for BLM = 0.

Target (%)	Algorithm	Min	Max	Median	Run time (s)
5	Greedy	113	113	113	1
5	Improved greedy	106	110	108	1
5	Simulated annealing	111	115	113	98
10	Greedy	208	208	208	1
10	Improved greedy	199	204	202	1
10	Simulated annealing	203	208	206	99

Table 2  
Simulated annealing results for BLM = 0 for varying annealing schedules.

Target (%)	$N$ in $10^3$ s	Min	Max	Median	Run time (s)
5	15000	111	115	113	98
5	1500	115	120	117	10
5	150	121	126	123	1
10	15000	203	208	206	99
10	1500	207	211	209	10
10	150	210	216	214	1

from running simulated annealing 10 times for each of three different values of  $N$  (with BLM = 0). These results show that as  $N$  increases, we correspondingly achieve lower values for the minimum and median number of sites required. The convergence aspect of simulated annealing appears to be well illustrated by this example. Hence, setting  $N$  to be very large, and ensuring an optimal annealing schedule is used, one would expect that simulated annealing would eventually match the solution given by the improved greedy algorithm.

The map in figure 2 indicates which sites have been selected by the improved greedy algorithm in the 5% representation problem, for a solution of 107 sites. It is visually obvious that the reserve system is very fragmented.

### 5.2. Nonzero BLM

In a bid to overcome this spatial fragmentation, nonzero BLMs are now considered. We are interested in the degree of compactness of reserve systems designed by minimising (IQP\*) for different values of BLM.

A simple measure of the degree of clustering among sites in a reserve system is the total boundary length of the reserve divided by the area. This is a measure of length per unit area. A more suitable dimensionless measure is the ratio of the boundary length of the reserve system to the circumference of a circle with the same area as the reserve [3,11], since a circle is the shape having smallest length for a given area, and is hence the “ideal” boundary length. The formula for this measure is

$$\text{compactness ratio} = \frac{\text{boundary length}}{2\sqrt{\pi \times \text{area}}}. \quad (5)$$

We use this measure of compactness to show that minimising  $C(\mathbf{x})$  for increasing values of BLM does indeed decrease the compactness ratio of a reserve design towards the ideal value for a given area.

Results were obtained using the greedy algorithm and simulated annealing, for values of BLM varying from 0.001 to 1000. Due to the nature of simulated annealing, the two easiest ways to try and find better solutions are increasing the number of neighbouring solutions considered in each run,  $N$ , or increasing the number of times the algorithm is run. The run time increases linearly in either case. For BLM zero, increasing  $N$  to a larger number was more efficient at finding better solutions than increasing the number of runs, since the spread of values was quite small, as can be seen in table 2. However, for nonzero BLM, we found that ten runs was insufficient to get a good spread of solutions for a given BLM and value of  $N$ . Hence, one hundred runs of each algorithm at each value of BLM were made and the minimum value of  $C(\mathbf{x})$  amongst these 100 runs is recorded in table 3 (5% target areas) and table 4 (10% target areas). Tables 3 and 4 also list the area (in units of the number of reserved sites), overall boundary length, BL, (one unit of BL represents the length of one side of a site) and compactness ratio, CR, for the reserve system with minimum  $C(\mathbf{x})$ .

Simulated annealing has in all but one case (marked with a \*) given better results than the greedy algorithm for minimum  $C(\mathbf{x})$ ; in many cases, substantially better results.

Note also that the compactness ratio decreases as BLM increases. For small values of BLM, the area is weighted more heavily than boundary length, and hence the algorithms try to select small numbers of sites. As BLM increases, a balance between area and length is achieved and clustering increases. With very large BLMs, the algorithms

start to include many sites which provide no value in terms of biodiversity, but which reduce the overall boundary length,

Table 3  
5% representation problem.

BLM	Greedy algorithm				Simulated annealing			
	$C(\mathbf{x})$	Area	BL	CR	$C(\mathbf{x})$	Area	BL	CR
0.001	113.4	113	412	10.9	111.4	111	430	11.5
0.01	117.1	113	412	10.9	113.2	109	420	11.3
0.1	154.2	113	412	10.9	147.4	114	334	8.8
1	425	159	266	6.0	354	130	224	5.5
5	1482	162	264	5.9	1232	142	218	5.2
10	2775	155	262	5.9	2282	222	206	3.9
20	5400	160	262	5.8	4329	409	196	2.7
50	11634	1734	198	1.3	9892	1092	176	1.5
100	21534	1734	198	1.3	18692	1092	176	1.5
1000	199734	1734	198	1.3	177092	1092	176	1.5

Table 4  
10% representation problem.

BLM	Greedy algorithm				Simulated annealing			
	$C(\mathbf{x})$	Area	BL	CR	$C(\mathbf{x})$	Area	BL	CR
0.001	205.7	205	678	13.4	207.8*	207	760	14.9
0.01	211.8	205	678	13.4	211.4	204	736	14.5
0.1	267.4	213	544	10.5	249.8	212	378	7.3
1	633	279	354	6.0	529	265	264	4.6
5	2050	340	342	5.2	1563	293	254	4.2
10	3617	1637	198	1.4	2803	643	216	2.4
20	5597	1637	198	1.4	4871	1191	184	1.5
50	11485	1685	196	1.3	10334	1234	182	1.5
100	21285	1685	196	1.3	19434	1234	182	1.5
1000	197695	1695	196	1.3	183234	1234	182	1.5

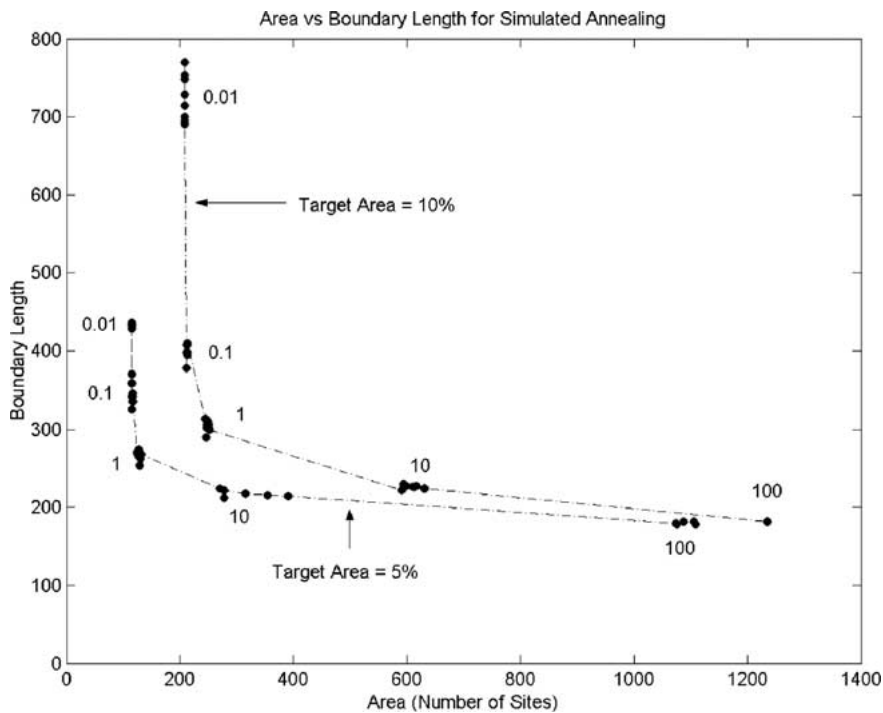


Figure 1. Plot of number of sites selected against boundary length for the 5% area target, and 10% area target. The numbers on the graph indicate values of BLM.

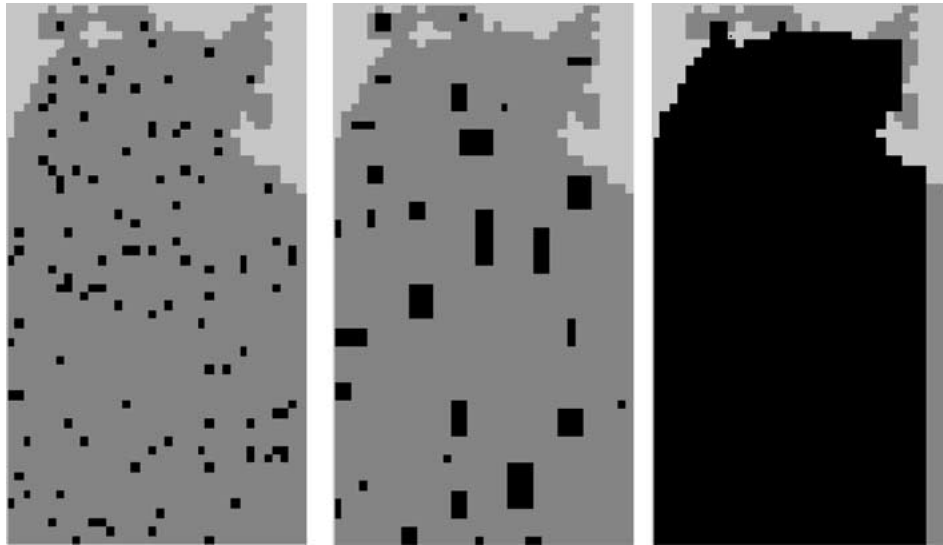


Figure 2. Maps of the Northern Territory showing reserved sites for the 5% area target, for the greedy algorithm with BLMs of 0 (left), 10 (centre) and 100.

until a point is reached at which no further decrease in the total boundary length is made. This point indicates the transition from a number of medium sized reserve clusters to one extremely large one. These trends are illustrated by figure 1, which was obtained using the best 20 results for each of a number of values of BLM. There is an interesting threshold between a BLM of 1 and 10 where we move from situations where area minimisation dominates, to where boundary length minimisation dominates. For a manager who essentially wants to minimise the area of the reserve system, but wants some clustering, then a BLM of just below 1 would appear to be a good choice for this data set. As we move the boundary length modifier above 1 there can be a rapid increase in reserve area. While this choice of BLM will depend upon the problem (the actual value of BLM that gives a particular solution is scale dependent, that is, it is dependent on the units used for length and area), we found the same effect for quite a different problem [3].

Maps of the Northern Territory indicating which sites have been reserved by the greedy algorithm for BLMs of 10 and 100 in the 5% area problem are shown in figure 2. Figure 3 shows the reserved sites for simulated annealing for BLMs of 1 and 10. It is visually clear that with a BLM of 10, the reserved sites are well clustered, and very few reserved sites have no neighbouring reserved sites. With a BLM of 100, nearly all sites have been reserved, reflecting the fact that boundary length is weighted much more heavily than area in the objective function.

It is interesting to note that for a BLM of 10, simulated annealing has resulted in larger clusters than the greedy algorithm, and less regular cluster shapes, for solutions not that dissimilar in total boundary length and area. This is a manifestation of the different way each algorithm builds a solution for this particular data set.

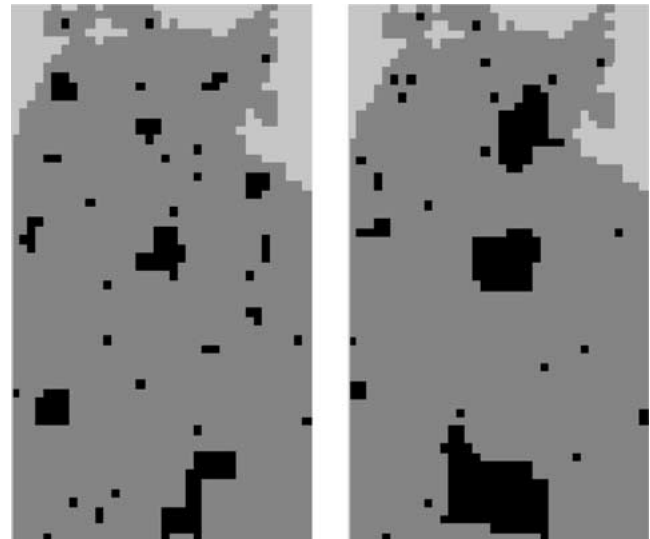


Figure 3. Maps of the Northern Territory, showing reserved sites for the 5% area target, for simulated annealing, with BLMs of 1 (left) and 10.

## 6. Conclusions

For the data considered in this paper, simulated annealing and the greedy algorithm both gave approximate optimal solutions to the bi-objective problem of minimising total reserve area and minimising total boundary length. By varying the value of the boundary length modifier, larger or smaller degrees of clustering were obtained. The simulated annealing algorithm usually gave better results in terms of minimising the combined cost function, although the greedy algorithm was much quicker to run and easier to implement. However, the run time for simulated annealing is not excessive and thus it appears that simulated annealing is the better method to use to decrease the cost or improve the clustering in a reserve design. Political, economic or other factors not

explicitly addressed in the model may determine which of a range of solutions is the best one to use.

Further research on this type of problem could focus on several areas: (a) more complex spatial requirements; (b) different biodiversity targets; and (c) a fixed amount of land available to be set aside. Another set of interesting problems arises when we attempt to simultaneously optimise multiple services (for example where a species has one value, say for ecotourism, when it is in a reserve, and a different value, say for trophy hunting, when it is outside a reserve). Solutions to these problems can provide conservation planners and decision makers with information that can be used to provide better protection for biodiversity.

## References

- [1] J.A. McNeely, K.R. Miller, W.V. Reid, R.A. Mittermeier and T.B. Werner, *Conserving the World's Biodiversity* (IUCN Publication Services, Gland, 1990).
- [2] M.E. Soule, Conservation: Tactics for a constant crisis, *Science* 253 (1991) 744–750.
- [3] H. Possingham, I. Ball and S. Andelman, Mathematical methods for identifying representative reserve networks, in: *Quantitative Methods for Conservation Biology*, eds. S. Ferson and M. Burgman (Springer, New York, 2000) pp. 291–305.
- [4] H.P. Possingham, J.R. Day, M. Goldfinch and F. Salzborn, The mathematics of designing a network of protected areas for conservation, in: *Decision Sciences: Tools for Today. Proceedings of 12th National ASOR Conference*, eds. D.J. Sutton, C.E.M. Pearce and E.A. Cousins (ASOR, Adelaide, 1993) pp. 536–545.
- [5] H.A. Taha, *Operations Research, An Introduction*, 5th Ed. (MacMillan Publishing Company, 1992).
- [6] R.L. Pressey, H.P. Possingham and J.R. Day, Effectiveness of alternative heuristic algorithms for identifying indicative minimum requirements for conservation reserves, *Biological Conservation* 80 (1997) 207–219.
- [7] I. Ball, A. Smith, J.R. Day, R.L. Pressey and H.P. Possingham, Comparison of mathematical algorithms for the design of a reserve system for nature conservation: An application of genetic algorithms and simulated annealing, *Journal of Environmental Management* (1998), in press.
- [8] R.L. Pressey, H.P. Possingham and C.R. Margules, Optimality in reserve selection algorithms: When does it matter and how much?, *Biological Conservation* 76 (1996) 259–267.
- [9] W.F. Fagan, R.S. Cantrell and C. Cosner, How habitat edges change species interactions, *American Naturalist* (1999), in press.
- [10] A.O. Nicholls and C.R. Margules, An upgraded reserve selection algorithm, *Biological Conservation* 64 (1993) 165–169.
- [11] I.R. Ball, Mathematical applications for conservation ecology: The dynamics of tree hollows and the design of nature reserves, Ph.D. thesis, The University of Adelaide (2000).
- [12] J.L. Cohen, *Multiobjective Programming and Planning* (Academic Press, 1978).
- [13] J. Wright, C. ReVelle and J. Cohen, A multiobjective integer programming model for the land acquisition problem, *Regional Science and Urban Economics* 13 (1983) 31–53.
- [14] K.C. Gilbert, D.D. Holmes and R.E. Rosenthal, A multiobjective discrete optimization model for land allocation, *Management Science* 31 (1985) 1509–1522.
- [15] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).
- [16] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, Optimisation by simulated annealing, *Science* 220 (1983) 671–680.
- [17] R.H.J.M. Otten and L.P.P.P. Van Ginneken, *The Annealing Algorithm* (Kluwer Academic Publishers, 1989).
- [18] M. Lundy and A. Mees, Convergence of an annealing algorithm, *Mathematical Programming* 34 (1986) 111–124.