

SPOT: The Spatial Portfolio Optimization Tool

Dan Shoutis

Copyright © 2003 by The Nature Conservancy

Contents

Acknowledgments	7
1 Overview	9
1.1 Introduction	9
1.1.1 History	9
1.1.2 Rationale	9
1.1.3 Technology	10
1.1.4 About this documentation	10
1.2 How it works	10
1.3 Limitations	11
2 Methodology: The SPOT Cost Function	13
2.1 Overview	13
2.2 Analysis units	14
2.3 Targets and goals	15
2.4 Target distribution	16
2.5 The portfolio boundary	16
2.5.1 The boundary length modifier	17
2.6 Target representation	18
2.7 Shortfall penalties	21
2.8 Example cost function	21
2.8.1 Analysis units	21
2.8.2 Boundaries	22
2.8.3 Targets and distributions	22
2.8.4 Calculating the cost function	22
3 Methodology: Simulated Annealing	25
3.1 Overview	25
3.1.1 Annealing	25
3.2 The mystery function	25
3.3 Hill descent	26
3.4 Simulated annealing	27
3.5 Considerations	29
3.5.1 Iterations	29

3.5.2	Balance	29
3.6	Evaluating an annealing run	30
4	Using SPOT	33
4.1	Installing	33
4.2	Preparing inputs	34
4.2.1	Choosing analysis units	34
4.2.2	Creating hexagonal analysis units	34
4.2.3	Setting analysis unit status	35
4.2.4	Preparing the target list	36
4.2.5	Importing target distributions	36
4.2.6	Editing target distributions	37
4.2.7	Creating boundaries	37
4.2.8	Viewing and switching inputs	38
4.2.9	Working with old SITES input files	38
4.3	Running the core	38
4.3.1	Just creating input files	38
4.3.2	Distributing SPOT runs over multiple computers	40
4.4	Viewing results	41
4.4.1	Loading a result	41
4.4.2	Summing results	41
4.4.3	Editing results	41
4.4.4	Reporting on a result	41
4.4.5	Reports	42
A	Table and theme formats	43
A.1	Inputs	43
A.1.1	Analysis units	43
A.1.2	Targets	43
A.1.3	Unit boundaries	43
A.1.4	Target distributions	44
A.1.5	Generated core inputs	44
B	SPOT internals	45
B.1	Annealing	45
B.1.1	The annealing process	45
B.1.2	Initial values	45
B.1.3	Generating a new portfolio	46
B.1.4	Evaluating the new cost	46
B.1.5	The acceptance chance of a portfolio	46
B.1.6	Updating the temperature	47
C	System Requirements	49
C.1	Hardware Requirements	49
C.2	Software Requirements	49

<i>CONTENTS</i>	5
D Glossary	53
Bibliography	55

Acknowledgments

This project would not have been possible without the support of Wayne Ostlie and the resources of The Nature Conservancy.

The SPOT methodology borrows heavily upon the research and development of the SITES portfolio selection tool by Hugh Possingham and Ian Ball at the University of Adelaide.

Special thanks to Pat Comer, Dan Dorfman, Mark Goering, Michael Heiner, Cherie Moritz, Betsy Neely, Dave Theobald, and Stephen Yanoff for their input and feedback during the planning and development of SPOT.

Chapter 1

Overview

1.1 Introduction

The Spatial Portfolio Optimization Tool (SPOT) is a generalized tool for conservation portfolio selection, using a flexible approach to automatically design an efficient portfolio around specified conservation goals.

1.1.1 History

SPOT is preceded by the SITES ecoregional planning tool, developed by the University of California at Santa Barbara for The Nature Conservancy (TNC) and used successfully by TNC for many ecoregional planning efforts.

SPOT provides the same functionality as SITES, but with increased transparency, faster run times, an easier to user interface, and greater extensibility. Although SPOT uses the same methodology as SITES, with a few minor differences¹, it has been developed from scratch using publicly available resources and shares no code with SITES.

1.1.2 Rationale

SPOT was developed with several goals in mind:

- Improve the user interface and make the methodology easier to use.
- Increase the transparency of the portfolio assembly process through additional reporting, more specific documentation, and source code.
- Create a SITES replacement with potential for future research, growth, and adaptation to new scenarios.

¹The largest differences between SPOT and SITES are that SPOT lacks a spatial separation rule for targets (a feature that has rarely been used with SITES for ecoregional portfolio assembly) and SPOT performs only simulated annealing, without the option of greedy heuristic or hill-climbing searches for a portfolio.

1.1.3 Technology

SPOT consists of a user interface extension for ArcView 3.x and a core program written in C++ and the Ruby scripting language. The user interface is used to manage input and output data, while the core program performs the actual portfolio assembly. See Appendix C for system requirements.

1.1.4 About this documentation

This guide is intended to be the authoritative reference for SPOT, and can be quite technical. This overview chapter should suffice to quickly gain a general understanding of the tool, but active practitioners should be familiar with the methodology and techniques described later on.

1.2 How it works

SPOT analyzes a region by dividing it into small parcels called analysis units, then forming a portfolio by marking individual units as included or excluded from a portfolio.

analysis unit: Analysis units form the portfolio; they are small areas that are marked as in or out to create a portfolio.

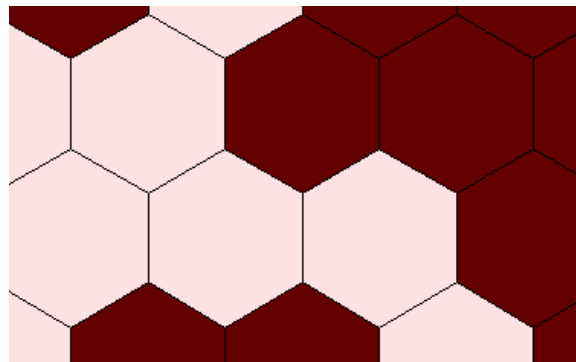


Figure 1.1: Detail of hexagonal analysis units in a SPOT portfolio; dark units are in the portfolio and light units were not included.

During a process known as simulated annealing, SPOT forms and analyzes millions of portfolios while searching for the most efficient portfolio. Each is evaluated according to three criteria:

simulated annealing: A general technique for finding the lowest value of a function through many trial runs and repeated adjustment to input values.

- How well it meets conservation goals
- The area included
- The fragmentation of the portfolio

The portfolio that does the best job of minimizing the area and fragmentation while meeting conservation goals is considered the most optimal, and is output as the final result.

SPOT also produces a number of reports that allow users to easily see the formation of a portfolio and the internals of the tool, as well as how well a portfolio meets target goals.

Additionally, there are number of parameters that can be changed and weighted differently, giving SPOT a broad range of flexibility and allowing a planning team to adjust SPOT to for different situations.

1.3 Limitations

- SPOT is prone, just as any other automated tool, to the “Garbage In - Garbage Out” syndrome. Any results will only be as good as the input datasets. Additionally, although SPOT is very flexible, the final portfolios it produces will be the best according only to its internal criteria, which may differ from what planners have in mind.
- SPOT creates and evaluates an entire portfolio at a time, which means that it never makes a decision about any individual area. Thus, there is no information available as to why specific areas were included or excluded from resulting portfolio.
- Although SPOT’s algorithm is statistically likely to find the most efficient portfolio given enough iterations, just how many iterations are necessary is variable and can change drastically with the nature of the region. Some experimentation will prove necessary.
- SPOT’s portfolio assembly algorithm is not deterministic. Running the tool multiple times on the exact same data will produce many, slightly differing, result portfolios. To overcome this limitation, planners should be performing many runs on the same data to ensure that SPOT is giving comparable results for each.

Chapter 2

Methodology: The SPOT Cost Function

2.1 Overview

SPOT attempts to assemble a portfolio with the minimal possible value of a cost function that encapsulates desirable characteristics for an ecoregional plan. The SPOT cost function is derived from the following goals for a portfolio:

- The portfolio should minimize the area required to adequately represent targets.
- The portfolio should meet conservation target goals set for the region.
- Fragmentation should be avoided; when choosing between a scattered area and a contiguous one with similar representation and size, the contiguous one is preferable.

In order to search for a portfolio that meets these principles, The region is first broken into small units known as analysis units. SPOT forms a portfolio by marking analysis units as included or excluded from the portfolio. Conservation goals are specified on a per-target basis, and each unit is attributed with the amounts of each target that it contains.

To take into account the three principles, the cost function is a sum of:

- A base cost for each analysis unit included in the portfolio. This will increase the value of the cost function as more analysis units are added, encouraging SPOT to find solutions that use less units.
- A shortfall cost, penalizing the portfolio for failures to meet goals. Every unmet target will increase the cost function's value.

cost function: The cost function for SPOT calculates a single cost value for a given portfolio that represents its effectiveness.

- A boundary cost the boundary of the portfolio. The way SPOT measures a portfolio's fragmentation is by the length of its boundary, and longer boundaries mean a higher value of the cost function.

base cost: A component of the SPOT cost function that encourages SPOT to minimize the area of the portfolio. It is the sum of the cost specified for each analysis unit included in the portfolio.

boundary cost: A component of the SPOT cost function, aimed toward minimizing a portfolio's fragmentation by minimizing the length of its boundary.

shortfall cost: A component of the SPOT cost function that penalizes portfolios that don't meet conservation goals.

More formally:

$$cost(x) = base(x) + boundary(x) + shortfall(x) \quad (2.1)$$

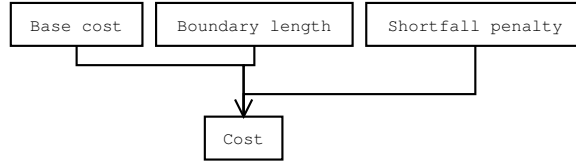


Figure 2.1: The cost function

2.2 Analysis units

Analysis units can be arbitrarily shaped and sized, depending on the needs of planners. In The Nature Conservancy's ecoregional plans, small hexagons have been used most often.

The prime consideration when choosing analysis units is size:

- If the units are too small, there will be so many of them that portfolio assembly will be unacceptably slow or fail to produce a robust answer.
- If the units are too large, the analysis will be too coarse and fail to adequately represent reality.

It is important to remember that everything in SPOT is based around analysis units: target distributions or other spatial information, no matter how fine-grained, is rounded up to the nearest analysis unit, much like information in digital images is rounded up to the nearest pixel. See Figure 2.5 for an example.

Analysis units consist of:

- An ID
- A base cost

When a unit is included in the portfolio, its cost is added to the total value of the cost function. This cost can represent simple area, or planners can use more sophisticated values to make some units preferred over others. Often, measures of GIS suitability are integrated into the basic unit cost.

The total base cost is thus:

$$base(x) = \sum_{k=1}^n BaseCost_k \quad (2.2)$$

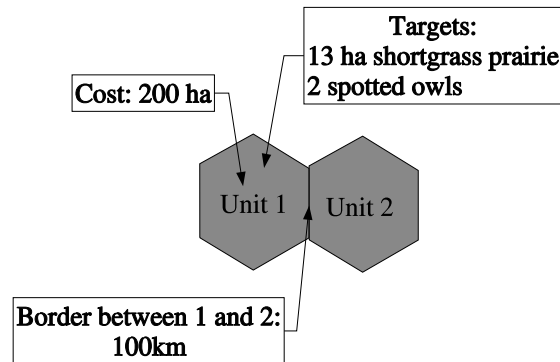


Figure 2.2: Analysis units, boundary information, and target distribution information.

for each unit k included in the portfolio.

2.3 Targets and goals

SPOT represents planners' goals by a list of targets, each representing a separate biological value that needs to be preserved. A target consists of:

- A numeric ID
- A name
- A goal
- A penalty factor
- A minimum representative area

For example, if a portfolio goal is to preserve at least 10,000 ha of a shortgrass prairie system type, then typical settings in SPOT's target table might be an ID of 1001, a name of "prairie", a goal of 10,000, a penalty factor of 1.0 and a minimum representative area of 0 ha.

SPOT will attempt to find portfolios that contain enough of a target to meet its goal. Portfolios that cannot fully represent a target will be penalized with a shortfall cost. (This calculation is discussed in more detail in Section 2.7.)

The minimum area requirement prevents SPOT from counting an occurrence of a target unless its contiguous size is greater than the specified amount. For example, if the type of prairie in the above sample target is only viable in occurrences of 1,000 ha or greater, then a minimum area requirement of 1,000 ha will force SPOT to collect connected analysis units that represent more than

target: A biological feature with a conservation goal that SPOT attempts to meet during assembly.

minimum area: The minimum contiguous amount of a target required for it to contribute to a conservation goal.

this amount before they can contribute to this target's goal. See Section 2.6 for more information on how this is calculated.

SPOT also allows planners to give a target a variable degree of impact on the portfolio via the penalty factor. Any shortfall penalties generated by a target are multiplied by this factor before being added to the total portfolio cost, so planners can weigh individual targets appropriately.

It is important to note that this weighting is relative to both the overall base and boundary components of the total cost, as well as other targets. (See Section 3.5.2.)

target patch: An occurrence of that can spread over several neighboring analysis units.

penalty factor: Sets the importance representing a target, relative to targets and the base and boundary components.

2.4 Target distribution

SPOT uses a table of target distributions to represent the spread of a target through a region, identifying each analysis unit with the targets that occur there and the amount of each:

- The analysis unit ID
- The target ID
- The amount of the target contained within the analysis unit

For the target distribution (with 1=Spotted Owl and 2=Shortgrass Prairie) given in Figure 2.2, SPOT's target distribution table will look like:

Target ID	Unit ID	Amount
1	1	13
2	1	2

2.5 The portfolio boundary

boundary length: The length of a portfolio's perimeter.

In order to discourage portfolio fragmentation, SPOT takes into account the portfolio's perimeter. (A fragmented portfolio will have a much longer boundary than a well-connected portfolio.)

SPOT represents boundaries with a table that contains the spatial relationship between neighboring analysis units.

Boundaries consist of:

- Two neighboring analysis unit IDs
- The length of the boundary shared between the units

To calculate the length of portfolio boundary, SPOT looks at every boundary between two units. If both units are in the portfolio, then that boundary is interior to the portfolio and not exposed. If one unit is in the portfolio, and the other out, the boundary is exposed and the indicated length is added to the total. Figure 2.3 illustrates this process.

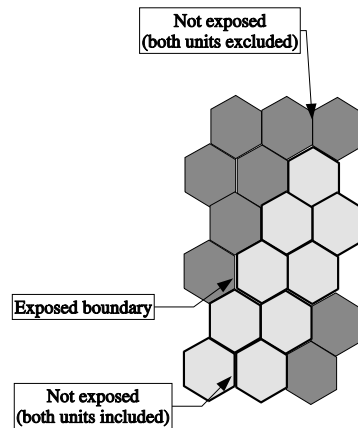


Figure 2.3: Finding exposed boundaries to calculate the boundary length of the portfolio.

Additionally, a unit can be specified as having a boundary with itself, which is useful for units that will always have a boundary when included in the portfolio (e.g. a unit at the edge of the region).

Before combining the calculated boundary length with the rest of the cost function, it is multiplied by a coefficient called the boundary length modifier, or the BLM.

boundary length modifier: A multiplier that converts and scales the boundary length of a portfolio before adding it to that portfolio's cost.

2.5.1 The boundary length modifier

Before the boundary length of a portfolio is added to the SPOT cost function (as the boundary cost), it is scaled by a factor called the boundary length modifier (BLM). The BLM serves several purposes, and is thus somewhat confusing:

- To specify the relative importance of fragmentation in the cost function. Smaller values will make fragmentation less important than meeting goals and minimizing area.
- To convert units. If the base analysis unit cost is specified as hectares (or, even more confusingly, GIS Suitability indexes), and boundary lengths as kilometers, the BLM must serve the purpose of converting the boundary into comparable units.
- To make “area” and “length” comparable. The least fragmented shape possible is a circle, and the area to circumference ratio can serve as a guide for this.

Because of the many conflicting factors inherent in the BLM, the best way to arrive at a good number is via experimentation.

The boundary portion of the cost function is:

$$\text{boundary}(x) = (BLM)\left(\sum_{k=1}^n \text{boundary}_k\right) \quad (2.3)$$

for every exposed boundary k .

2.6 Target representation

SPOT calculates the representation of a target in a portfolio in the following way:

If the target has no minimum representation requirements:

- For every unit that is marked as being in the portfolio and contains the target, the amount is added to come up with a total.

If the target has a minimum representation, SPOT uses the following procedure to take into account the size of target patches before adding them to the total representation (Figures 2.4 and 2.5 illustrate this process):

- SPOT begins by finding a unit that is in the portfolio with the target present.
- SPOT adds the amount to a temporary running total.
- For every unit that: shares a boundary with the current unit, is in the portfolio, contains some of the target, and has not already been examined; SPOT adds the amount to the running total.
- SPOT then repeats this process with the neighbors' neighbors, then those units' neighbors, and so on, until it runs out of connected units that contain the target.
- If the running total of this connected patch is greater than the target's minimum area requirement, then SPOT adds it to the target representation amount. Otherwise, it is dropped.
- SPOT continues finding patches in the way described above until the entire portfolio has been examined.

It is important to note that the accuracy of this procedure depends to a large extent on analysis unit size: If two neighboring units have presence records for the same target, it is assumed that they are part of a larger presence that covers both. This may not be the case, since neighboring units may have two independent occurrences that will be erroneously added together during the minimum area assessment (Figure 2.5). One way to avoid this pitfall is to drop all occurrences that fall below the minimum area (such as the small patch in unit 6 in the figure) as a preprocessing step, before entering them into SPOT's target distribution table.

SPOT uses the target representation to calculate a shortfall penalty, as well as to report back to planners on goal performance.

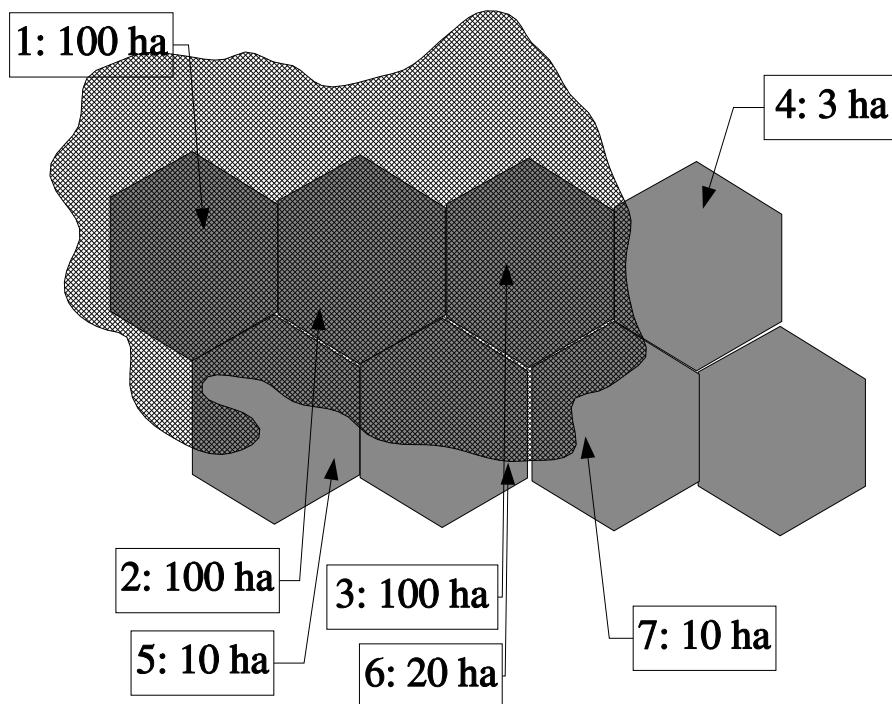


Figure 2.4: **Minimum Representation in action:** The distribution of a target with minimum representation. If the minimum representation for this target was 110 ha, then a portfolio that included units 1 and 2 would meet the requirement and contribute 200 ha toward the target goal. A portfolio that included only units 1 and 6 would not have a connected target presence sufficient to meet the representation requirement, and so there nothing would be contributed toward the target goal.

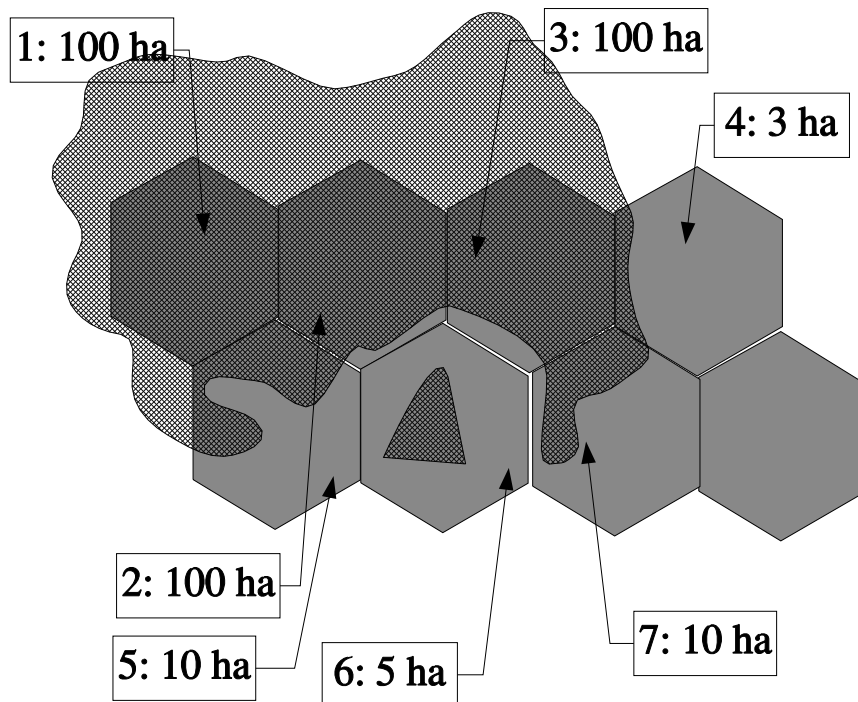


Figure 2.5: **Issues with minimum representation and analysis unit size:** With a minimum representation requirement of 110, a portfolio that includes units 1, 5, and 6 will meet the minimum necessary area. Even though the actual target distribution is not connected, SPOT will assume that the 5 ha in unit 6 are connected to those in 5 and 1 because the analysis units are touching.

2.7 Shortfall penalties

In order to calculate a penalty for portfolios that fail to represent targets, SPOT first pre-calculates an initial penalty amount for each target, designed so that the penalty imposed for a shortfall approximates, and is slightly greater than, the cost required (in terms of base cost + boundary length) to fully represent the target. Appendix B describes this in more detail.

The practical effect of this is that the cost of making up a shortfall will be slightly less than the penalty imposed by the shortfall; this way the simulated annealing process will favor portfolios with more complete target representation.

Each penalty is calculated in an initial phase where SPOT builds a mini-portfolio for each target. The cost of this mini-portfolio, which is a good approximation of the cost to fully represent the target, is then stored as a penalty cost.

Shortfall penalties are calculated by multiplying this cost by the proportion of any shortfall, as well as a target's penalty factor.

$$\text{ShortfallPenalty} = (\text{PenaltyFactor}) \left(\frac{\text{ShortfallAmount}}{\text{GoalAmount}} \right) (\text{PenaltyCost}) \quad (2.4)$$

For example, if a goal is 90% met, the penalty cost will be 10% of the calculated amount to represent the full target.

Additionally, a target's penalty factor is multiplied by the initial shortfall cost to arrive at a final shortfall penalty for that target.

2.8 Example cost function

This section ties everything together with a simplified portfolio assembly situation, calculating the cost function for a portfolio by hand.

2.8.1 Analysis units

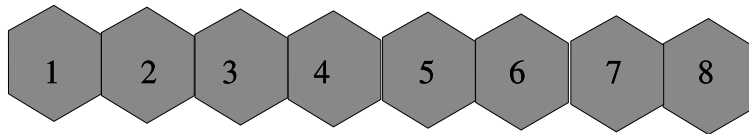


Figure 2.6: The analysis units for a simplified cost portfolio assembly situation.

The analysis units for this portfolio are specified in Figure 2.6, and the corresponding table of IDs and costs is:

Unit ID	Unit Cost
1	100
2	100
3	150
4	100
5	100
6	175
7	200
8	100

2.8.2 Boundaries

Each unit shares boundaries with one or two neighbors. If the side of each hexagon was 100 units long, the boundary definition table will be:

Unit A	Unit B	Length
1	2	100
2	3	100
3	4	100
4	5	100
5	6	100
6	7	100
7	8	100

2.8.3 Targets and distributions

This simple portfolio has targets “Trees and Grass” specified as follows:

Target Name	Goal	Penalty Factor	Penalty Amount
Trees	50	1	400
Grass	1000	1.5	200

The target distributions are:

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7	Unit 8
Trees	30	30	30	30	0	0	0	0
Grass	0	600	0	0	800	0	120	0

2.8.4 Calculating the cost function

For this example, assume that units number 1, 3, and 5 are included in the portfolio. What would the value of the cost function be with a BLM of 0.5?

The base cost

The base cost for the portfolio is the sum of the cost of each analysis unit:

$$basecost(x) = 100 + 150 + 100 = 350$$

The boundary cost

The boundary cost is the BLM, times the length of all exposed boundaries. The boundaries between 1 and 2, 2 and 3, 3 and 4, 4 and 5, and 5 and 6 are all exposed. This gives rise to:

$$\text{boundcost}(x) = 0.5 * (100 + 100 + 100 + 100 + 100) = 250$$

The shortfall cost

The shortfall depends on the target representation. Since neither target has a minimum representative area, the amount in the portfolio is the sum of all distributions in selected units.

Target	Amount
Trees	$\text{amount}(x) = 30 + 30 + 0 = 60$
Grass	$\text{amount}(x) = 0 + 0 + 800 = 800$

The shortfall cost imposed by each target is calculated as follows:

$$\text{ShortfallPenalty} = (\text{PenaltyFactor}) \left(\frac{\text{ShortfallAmount}}{\text{GoalAmount}} \right) (\text{PenaltyCost})$$

Target	Shortfall Cost
Trees	$\text{ShortfallPenalty} = (1.0) \left(\frac{0}{50} \right) (400) = 0$
Grass	$\text{ShortfallPenalty} = (1.5) \left(\frac{200}{1000} \right) (200) = 60$

$$\text{shortfall}(x) = 60$$

The overall cost

$$\text{cost}(x) = \text{base}(x) + \text{boundary}(x) + \text{shortfall}(x) = 350 + 250 + 60 = 660$$

There is an interactive Excel worksheet distributed with SPOT, called **SPOT-CostMockup.xls** that allows you to experiment with the cost function in this situation.

Chapter 3

Methodology: Simulated Annealing

3.1 Overview

Simulated annealing is the name for a general algorithm to find the general minimum value of a “mystery function” (Figure 3.1). Simulated annealing has proven to be an effective way of approaching many computationally difficult problems, including ecoregional portfolio assembly. In SPOT’s case, this algorithm is used to search for the portfolio that produces the lowest value of the cost function described in Chapter 2.

3.1.1 Annealing

The name is derived from the process of a slowly changing state in materials such as water freezing from a liquid into a solid. When the speed of the temperature drop is carefully controlled in order to arrive at a near-ideal final crystalline state, it is known as annealing.

annealing: The technique of slowly cooling a liquid into a solid such that its final form is a near-optimal crystal.

3.2 The mystery function

This metaphor is extended to the SPOT cost function, $f(x)$. If we treat x as a potential portfolio, then $f(x)$ is the value of the cost function for that portfolio. Because it is extremely difficult to predict anything about the impact of any given unit on the total portfolio cost, the SPOT cost function is used as the mystery function for simulated annealing.

Although SPOT uses simulated annealing with its own cost function, $f(x)$ could be any function that takes in a “state” and returns a single value we can refer to as “cost,” making annealing a very flexible technique for computationally daunting problems.

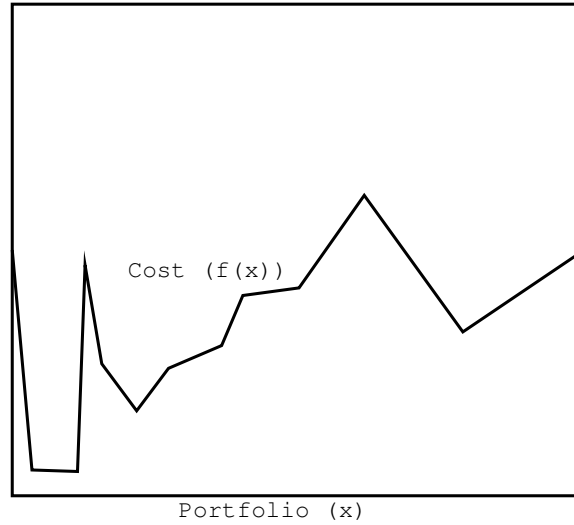


Figure 3.1: A mystery function

3.3 Hill descent

It would be impossible to evaluate every possible portfolio to determine the best configuration, since even a simple SPOT portfolio assembly problem with 300 analysis units would have

$$2^{300} \approx 2.037 * 10^{90}$$

combinations, which is more than most estimates of the number of atoms in the entire universe.

Instead, we can generate a starting portfolio randomly and evaluate $f(x)$ to determine the cost. From there, a simpleminded approach would be to generate another portfolio by making small random changes to the current one and accept them only if the changes made an improvement to the cost. This would continue for many iterations until no further improvements are possible.

$$x_{new} = randomchange(x_{current})$$

$$x_{current} = \begin{cases} x_{new}, & f(x_{new}) < f(x_{current}) \\ x_{current}, & f(x_{new}) \geq f(x_{current}) \end{cases}$$

This technique is called hill descent¹, and is closely related to simulated annealing. However, this algorithm will get stuck when it reaches a point where all small changes will result in higher cost, but a better solution exists elsewhere. This is called a local minimum. In Figure 3.2, B is a local minimum, while 4 is the absolute lowest minimum.

iteration: A single change to the current state and re-evaluation of the cost function.

local minimum: A low point in the cost function, but not the absolute lowest point.

¹This is usually called hill *climbing* in the literature, but the name has been changed to reflect our search for minimums rather than maximums.

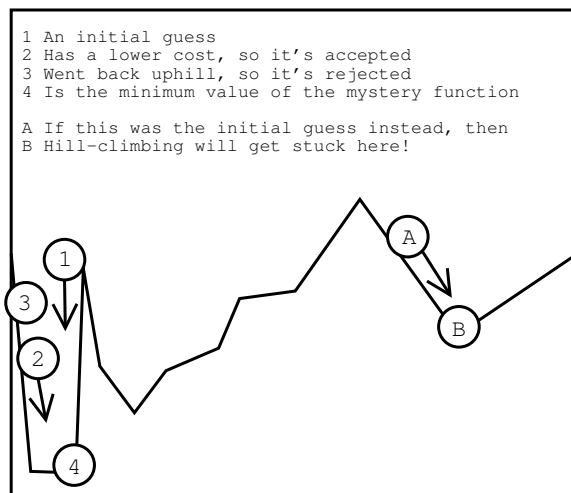


Figure 3.2: Hill descent.

One way to visualize the process of hill descent is to imagine the cost function as a landscape. Each position represents a different portfolio, and the elevation at that point represents the cost. The current portfolio is like a ball that can only roll downhill, which means that it will quickly seek the lowest nearby value. However if it starts in the wrong place, it will end up in a local minimum instead of the absolute minimum.

3.4 Simulated annealing

Simulated annealing is designed to avoid being stuck at a local minimum at the expense of increased runtime.

The simulated annealing process can also be imagined as a ball (the current solution) exploring the landscape, but rather than simply rolling downhill it bounces randomly. How the current portfolio changes (where the ball bounces) is driven by a factor called the temperature, which is analogous to the real-world temperature during the physical (non-simulated) process of annealing.

At first, when the temperature is high, the ball will cover large distances and cross over hills easily. (Figure 3.3) As the algorithm progresses, the temperature is lowered and the annealing process is less likely to accept large cost increases – the ball is less likely to make large jumps uphill. (Figure 3.4)

By lowering the temperature at an appropriately slow rate, depending on the size and steepness of the landscape, the solution has a much better chance of settling into the lowest possible point – much as slowly cooling a material will lead to an optimal crystalline structure.

temperature: In simulated annealing, this defines the maximum allowable change in the cost function

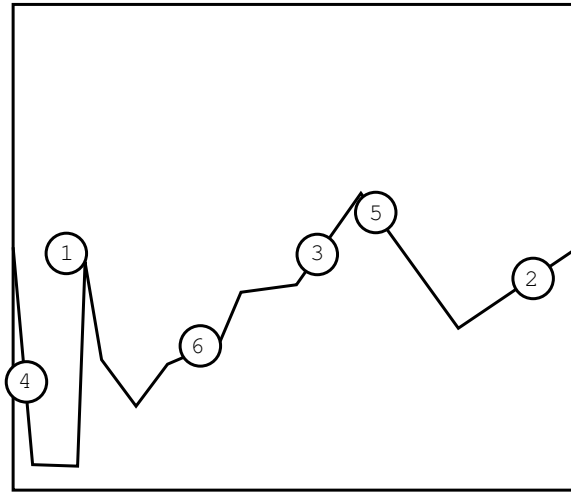


Figure 3.3: At high temperatures, large changes in the state and big steps uphill are acceptable.

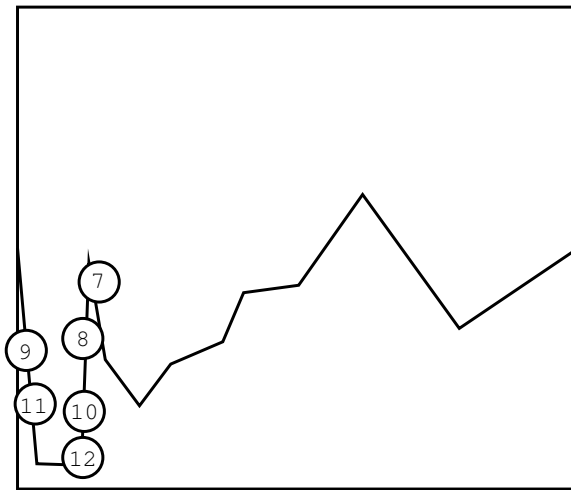


Figure 3.4: At lower temperatures, the changes are smaller and big steps uphill are rejected.

3.5 Considerations

3.5.1 Iterations

With this process of simulated annealing, the main concern is that enough iterations are specified that the cost function adequately explores possible portfolios and doesn't get stuck in a local minimum.

Defining "enough" can be tricky. Much depends on the complexity of the cost function: A function with very steep and tall features (Figure 3.5) where the cost often changes drastically with only a small change to the portfolio will take more iterations than one that changes smoothly and has a broad, well-defined minimum (Figure 3.6).

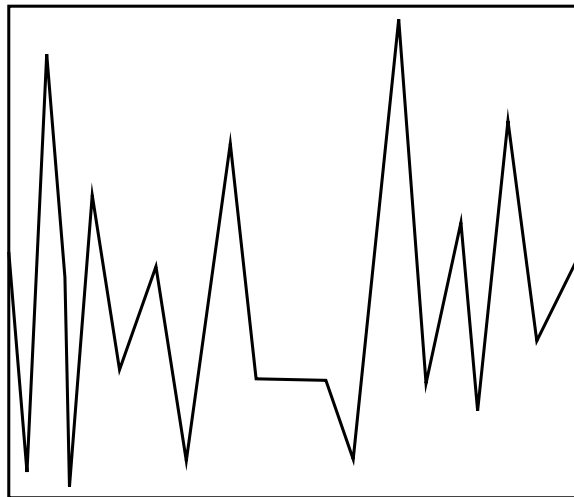


Figure 3.5: A difficult cost function

SPOT provides a number of outputs that provide some insight into the shape of the cost function and the performance of the simulated annealing algorithm, which can help diagnose a SPOT run (see Section 3.6).

Additionally, SPOT saves several (usually 10) interim portfolios during the annealing process, and these can be examined to see how a portfolio came together.

However, the best way to ensure that a SPOT isn't generating a local minimum is to do many runs with widely differing initial portfolios and verifying that they arrive at the same result.

3.5.2 Balance

If all three terms in the SPOT cost function are important to the final result, it's a good idea to be careful that their contribution to the portfolio cost is relatively equal. If any one term is weighted too heavily, the portfolios generated by SPOT

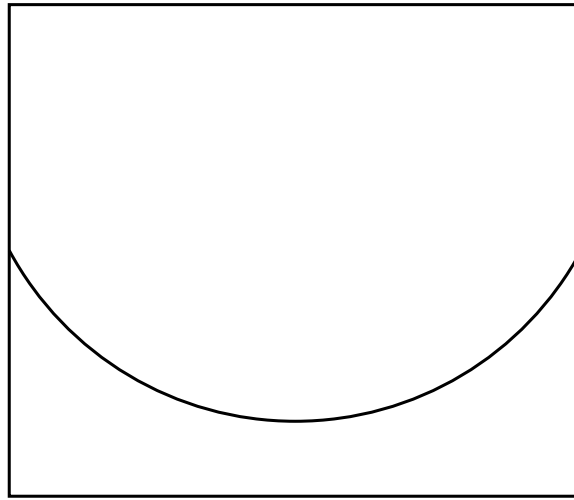


Figure 3.6: An easy cost function

may find a reasonable way of satisfying that term but never reach a solution that also performs well on the less significant factors. (This is also closely related to the difficulty of the cost function, since factors that are over-weighted will increase the function's sensitivity to small changes.)

3.6 Evaluating an annealing run

SPOT outputs a number of trace files that contain the value of internal variables, usually sampled every 100 iterations. Graphing these can provide much insight into the internals of SPOT's assembly process.

SPOT creates an HTML file in the log folder with these graphs, but to really explore them, you should use a plotting program that allows you to zoom and explore. Gnuplot, an open source graphing program, is included with and used internally by SPOT and can be used but has a steep learning curve. Excel can also be used to chart the logs (import as a space-delimited text file), but does not allow zooming.

Some of SPOT's logs are described below.

Cost

This log, named "CURCOST," contains the current value of the cost function throughout annealing. If it doesn't decrease through the run, then there is a problem, since the entire point of simulated annealing is to find the minimum cost.

The graph of this can give you an idea of the complexity of the cost function in your region – a smooth decrease with a homogeneous level of fluctuation that

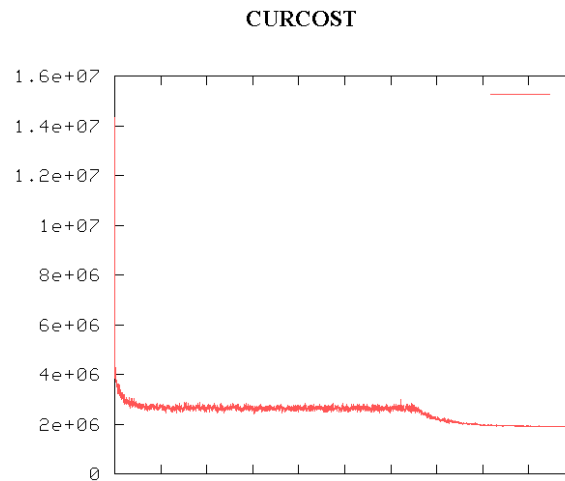


Figure 3.7: SPOT's log of the current portfolio's cost during a run.

also gets smaller toward the end of annealing means that your cost function worked fairly well, while a number of big jumps that stand out can represent a more difficult cost function.

Proposed cost

This log, "TRIED_COST," contains the proposed cost of the new portfolio at each iteration – not necessarily the one that was accepted, since SPOT will reject some portfolio changes, depending on the current temperature and how better/worse the cost function became.

Again, out-of-proportion jumps can signify a more difficult cost function.

Cost change

The "COST_CHG" log contains the change in the cost function at each iteration. Spikes in this plot indicate iterations where the current cost changed drastically; in some old SITES input files there were individual changes with values of more than two billion while the average change was invisible on the graph at just a few hundred.

If the value of the cost function can change so much (orders of magnitude larger than most other changes) between two portfolios where only a single analysis unit changed status, it is safe to say that it falls into the "difficult" category of cost functions.

Other logs

SPOT includes a number of other logs, which are described in Appendix B.

Chapter 4

Using SPOT

SPOT is split into two parts: the core, which performs the annealing process, and a user interface which helps organize and create input data before sending it to the core.

The user interface is an ArcView 3.x extension, with portfolio data stored in regular ArcView shapefiles and tables. It assists with creating and managing input data, as well as running the core and viewing results.

SPOT stores portfolio and input information alongside a view, you can use different view documents for different portfolios even within the same project.

core: The SPOT module that performs the annealing process to create a portfolio

4.1 Installing

To install SPOT, run the SPOTInstall.exe from the SPOT CD, or after downloading it, and follow the prompts. It should automatically install the user interface into the ArcView extension folder and the core to a location of your choice.

To begin using SPOT, start ArcView and enable the SPOT User Interface extension. Whenever you're in a View document, a SPOT menu will be visible.

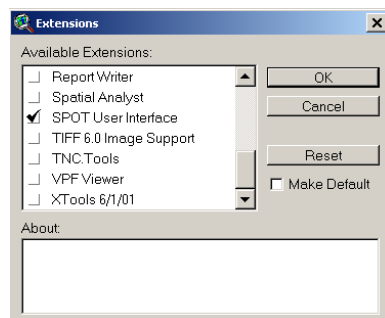


Figure 4.1: Enabling the SPOT ArcView extension

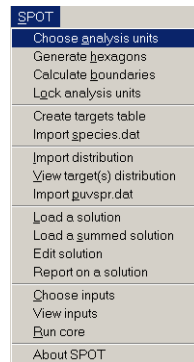


Figure 4.2: The SPOT menu

4.2 Preparing inputs

4.2.1 Choosing analysis units

Before you can do much else with SPOT, you'll need to define the analysis units to work with. Any shapefile theme can be used for analysis units, but non-polygon themes will limit SPOT's other functionality.

The theme you wish to use for analysis units must have:

- A numeric ID
- A field with the size of the unit
- A field with the base cost for the unit for the cost function

To select a theme to use as analysis units, use the **Choose analysis units** SPOT menu item and select the appropriate fields to use.

4.2.2 Creating hexagonal analysis units

SPOT comes with the ability to create hexagons that cover a given theme. To do this, use the **Generate hexagons** menu item. SPOT will ask for a theme to "hexagonize," then ask for the method of sizing hexagons:

- Auto-size: SPOT will size the hexagons so that it will generate approximately the specified number of hexagons. This is a loose approximation.
- Size hexagons by the length of a side: Each hexagon will have sides of the specified length; there will be enough of them to completely cover the input shapefile.
- Size hexagons by area: Each hexagon has the specified area.

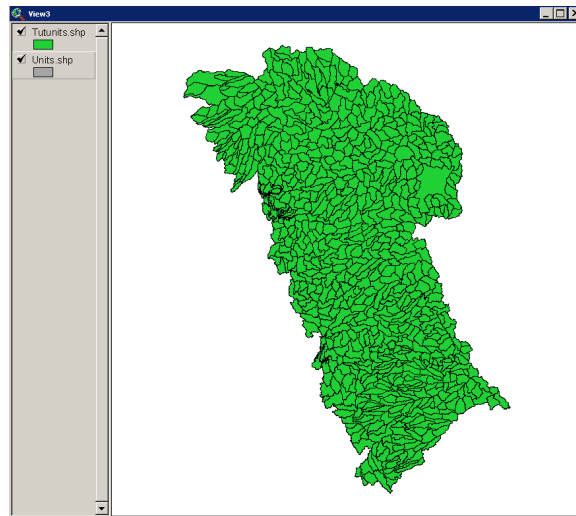


Figure 4.3: After analysis units have been chosen

SPOT will then ask for the number of hexagons, the length of a side, or the area of a hexagon, depending on which method was selected. Areas are measured in the view's current map units. Initially, the cost of each unit will be zero.

When finished, SPOT will add the new shapefile to the theme and use it for the analysis unit theme.

Analysis unit areas

SPOT will automatically deduce unit conversions for all automated measurements (e.g., while importing target distributions or creating boundaries) into the same units used for the analysis unit area field. This means that:

- Area measures will be in the same units as analysis unit areas
- Lengths are measured in the linear equivalent of the area unit (meters if the area is in square meters, feet if the area is square feet, 100-meter lengths if the area is in hectares, etc.)

4.2.3 Setting analysis unit status

After the analysis units have been chosen, users can change the status of the units for the annealing process. There are four options:

- Locked in: The selected units are locked into the portfolio – they must be part of the final solution, and all solutions along the way.
- Locked out: The selected units cannot be part of any solutions.

- **Initially in:** The selected units will be in the portfolio for the first iteration, but are not fixed after that.
- **Initially out:** The selected units will not be in the portfolio on the first iteration. This is the default in SPOT (and SITES).

To change the status of some units, select them using ArcView's selection tools and use the **Lock analysis units** command.

4.2.4 Preparing the target list

SPOT maintains the target list in an ArcView table. To create a blank target list, use **Create targets table**. To add new targets, simply add them to the table. If you want to maintain compatibility with SITES, you should make sure there are no spaces in the names of targets.

You can also create or import a targets table using the tool of your choice, and use **Choose inputs** to select it as the targets table. Appendix A has information on the necessary structure. This gives users the freedom to maintain information on the target list and planning process in a more convenient tool, such as Excel or Access.

4.2.5 Importing target distributions

SPOT is capable of importing target distributions from arbitrary ArcView shapefiles. The shapefile can contain the distribution of multiple targets, as long as they are identified by some ID. (If the IDs used don't match IDs from SPOT's targets list, SPOT will prompt you to identify the target that corresponds with each unrecognized ID.)

To import a distribution, use the **Import distribution** command. SPOT will ask for:

- The theme to import
- The field that contains IDs (these can be anything, so long as different targets use different IDs in the theme).
- For each ID found in the shapefile, SPOT asks which target (from the targets table) corresponds with it.
- For each target, whether this theme should replace any existing distribution or add to it.
- How each target should be measured during the import.

SPOT will automatically intersect the target distribution against the analysis units, perform bookkeeping, and convert units appropriately. (See 4.2.2 for a note about unit conversion).

There are several ways of tracking target distributions that SPOT supports. The behavior of SPOT's target distribution bookkeeping depends on what you select during the import:

- **Area:** SPOT keeps track of the area of the target present within each analysis unit. This option is only available if your distribution table is of polygons. If you do not delete pre-existing distributions, then the imported distribution will be added to the older one.
- **Length:** Similar to area, but for linear targets.
- **Count:** SPOT will count each individual record falling within an analysis unit. This is additive with older data unless you tell SPOT to delete the preexisting distribution first.
- **Presence/Absence:** SPOT will mark each analysis unit as either containing or not containing any of the target.

Distributions are stored in an ArcView table, created the first time you import a shapefile. Appendix A describes the structure of this file.

SPOT also supports target imports from integer grids, following the same process as above, with the following exceptions:

- Importing a target from a grid will always replace any existing data SPOT has about its distribution.
- Grid target imports are always measured by area.

4.2.6 Editing target distributions

After importing some target distributions, you can view the results by using **View target distribution** and selecting the desired target(s).

SPOT will create a shapefile showing what its internal table says about a target's distribution, containing:

- The analysis unit ID
- The target ID
- The target name
- The amount of the target

You can edit this distribution, adding or removing records and changing amounts, then use **Import distribution** to incorporate your changes. SPOT will recognize it and your changes to the distribution will be automatically updated in the full distribution table.

4.2.7 Creating boundaries

To calculate boundary length information, use the **Calculate boundaries** command. SPOT will create a new shapefile with boundaries between analysis units as lines, attributed with the IDs of both units and the boundary length (see 4.2.2 for information on units).

This shapefile can be edited if the boundary information needs to be revised.

Note that SPOT can only create boundaries for polygon analysis units. Also, this process can be slow for complex analysis units. SPOT can create boundaries for analysis unit themes that are somewhat sloppy (e.g., cracks between units). If you hold the **Shift** key down while choosing this menu item, you can modify the epsilon value for SPOT's tolerance of non-perfect input.

4.2.8 Viewing and switching inputs

You can keep several versions of input tables and themes around and use the **Choose inputs** command to select them for use. SPOT will verify their structure before selecting them.

The **View inputs** command will bring up the currently selected input tables.

This is also useful if you opt to create inputs by hand using the guidelines in Appendix A.

4.2.9 Working with old SITES input files

SPOT can automatically import target lists and distributions from SITES input files. **Import species.dat** will bring in a SITES target list, and **Import puvspr.dat** will bring in target distributions. The SITES manual describes the format of these files.

4.3 Running the core

Once the input tables and shapefiles are ready, use **Run core** to begin the annealing process.

SPOT will ask for a scenario name, the number of iterations and the value to use for the Boundary Length Modifier. Once the values are set, press **Go**.

SPOT will then prompt for the location where inputs should be created and results generated before running the core module.

Before the core module comes up on screen, it will generate target penalties, so there will be a few seconds before the core module appears.

To start the annealing process, press **Go**. SPOT will begin crunching and the computer may seem locked, but progress information will be periodically updated (generally 10 times per run).

4.3.1 Just creating input files

If you'd like to create input files without running the core, you can check the appropriate box (**Just create input files** at the bottom of the Run SPOT dialog). After clicking **Go**, SPOT will ask where they should be saved and take no further action.

Since SPOT maintains compatibility with old SITES input file formats, you can use this feature if you'd prefer to run SITES rather than the SPOT core.

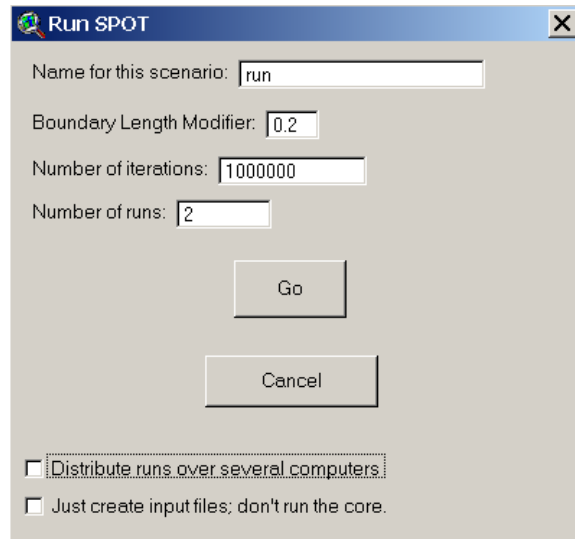


Figure 4.4: Running the core

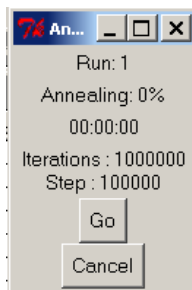


Figure 4.5: The core

4.3.2 Distributing SPOT runs over multiple computers

SPOT is capable of automatically distributing runs over several computers on the network, thus taking advantage of unused computer time and greatly reducing the time to receive results.

Taking advantage of this has the following drawbacks:

- A summed solution is not created. You can create this yourself by using **Load a summed solution** and manually adding in each run's result.
- Diagnostic logs are not produced.

Multi-computer SPOT is distributed in two parts: The master, which controls SPOT runs and is started from the primary SPOT workstation, and the slaves, which are distributed over the network.

Setting up a slave

To start a slave use the **SPOT Slave** item from the SPOT entry on the Windows start menu.

Depending on your network setup, the SPOT master may not be able to automatically discover all slaves, so you may need to write down the IP address of each slave.

On Windows 2000 and XP, you can find this number from the command prompt using the **ipconfig** command. On 98, use **winipcfg**.

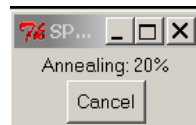


Figure 4.6: The SPOT slave program

ArcView does not need to be installed on any of the slave computers.

Starting the master

The SPOT master is started from ArcView by following the same steps to run SPOT normally, but checking the **Distribute runs over multiple computers** option box. (See Figure 4.4.)

The SPOT master program will come up, rather than the normal core.

After starting your SPOT slaves, you can click the **Connect slaves** button to connect them to the master and begin the annealing process.

For some network configurations, the SPOT master is unable to automatically locate and start slaves; in this case, type the IP address of a slave into the text box above the **Connect slaves** button before clicking to manually connect.

As with normal SPOT runs, the slave and master programs may seem frozen; but they will occasionally update.

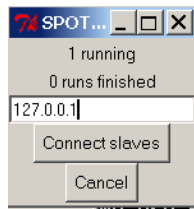


Figure 4.7: The SPOT master program

4.4 Viewing results

4.4.1 Loading a result

Once annealing has completed, you can load the final result by using **Load a solution**. An overall best solution, out of all runs performed, will be in the **outputs** folder in the location specified for the scenario, called **overallbest.txt**.

Additionally, inside the numbered folder for each run, there will be a file named **bestsol.txt**. This is the final solution for that run. Finally, SPOT creates 10 intermediate solutions per run, stored in the state folder for the run and named by the iteration.

Once you've picked the result to load, SPOT will bring it in as a new theme, similar to the analysis units theme, but with just a single field indicating the final result status (1=in, 0=out).

4.4.2 Summing results

SPOT can sum solutions over multiple runs, providing an easy way to tell how robust solutions are and how frequently units showed up in different solutions. The use of this is similar to loading a normal solution, only using **Load a summed solution**.

SPOT will prompt for at least one summed solution file, but planners can choose to add many together if they wish (including normal solution files), to bring together results from different scenarios. SPOT automatically creates a sum of all runs called **summedsol** in the **outputs** folder.

4.4.3 Editing results

Often, you'll want to tweak the results coming out of SPOT by changing the status of some analysis units.

To do this, select the units using the ArcView selection tool, then use **Edit Solution** to change their status in the result.

4.4.4 Reporting on a result

If you want to see the cost elements associated with an edited result, you can use **Report on a solution** to generate a report containing the target representation

and cost function factors for the result.

This is especially useful for determining the targets represented by a smaller subset of a portfolio. Because of minimum representation requirements, calculating the amount of a target present at a site can be complicated. By editing a solution, you can create a new portfolio consisting solely of the area you're interested in and generate a report to tell you what targets are represented there.

Note that these reports are generated using the current units, boundaries, targets and distributions – if you've changed them since you ran SPOT, the generated report will reflect your edits to the input data as well as the result.

4.4.5 Reports

SPOT creates a report in HTML format under the **outputs** folder for each run it completes. It contains sections for:

- Overview information
- Cost function breakdown
- Target representation

This file can be opened directly into Microsoft Excel for further analysis and formatting. (Users can also cut and paste tables into Excel MS Internet Explorer.)

Appendix A

Table and theme formats

This appendix has the table and theme formats for SPOT's internal shapefiles and tables. The size and precision of numeric fields is flexible, but a width of 32 digits and least 8 decimal places (for double-precision numbers) is recommended.

A.1 Inputs

A.1.1 Analysis units

Analysis units can be any shapefile that has the following fields:

Name	Type	Notes
id	Integer	This must be unique.
cost	Double	Contains the base cost for the unit.
area	Double	See 4.2.2 for how this affects SPOT's operation.
status	Integer	0 Initially in
		1 Initially out
		2 Locked in
		3 Locked out

A.1.2 Targets

Targets can be any ArcView table:

Name	Type	Notes
tgt_id	Integer	Unique ID for the target.
name	String	If you want compatibility with SITES, you shouldn't have any spaces in the name.
goal	Double	The target goal.
penfact	Double	The target penalty factor.
minarea	Double	The minimum representative area for the target.

A.1.3 Unit boundaries

SPOT stores unit boundaries in a shapefile:

Name	Type	Notes
id_a	Integer	Identifies one of two neighboring analysis units.
id_b	Integer	Identifies the other unit.
length	Double	The length of the boundary between the units.

A.1.4 Target distributions

Target distributions are stored in a table with the following format:

Name	Type	Notes
pu_id	Integer	Identifies an analysis unit.
tgt_id	Integer	Identifies a target.
Amount	Double	How much of the specified target is present at the analysis unit.

A.1.5 Generated core inputs

In the **inputs** folder for a run, SPOT generates input files that correspond with the various user interface tables. These are compatible with the input file formats used for SITES, and their structure can be looked up in the SITES documentation.

SPOT generates:

- name.dat
- cost.dat
- pustat.dat
- species.dat
- puvspr.dat
- bound.dat

Appendix B

SPOT internals

B.1 Annealing

B.1.1 The annealing process

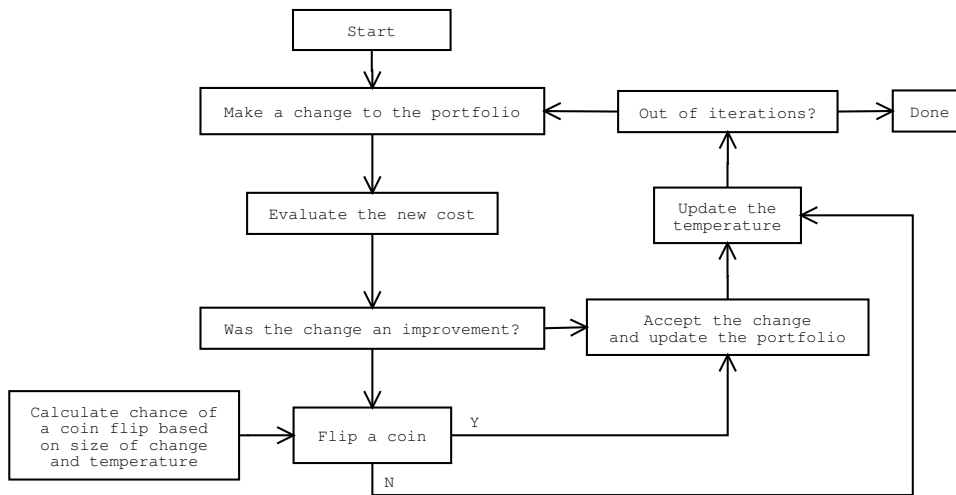


Figure B.1: SPOT's simulated annealing algorithm

B.1.2 Initial values

- The current portfolio is set according to the initial analysis unit status.
- The temperature is equal to the average change in cost induced by flipping random 100 analysis units in and out of the portfolio.
- The acceptance rate is set to 50%.

- Target penalties are calculated. (See Figure B.2.)

Calculating target penalties

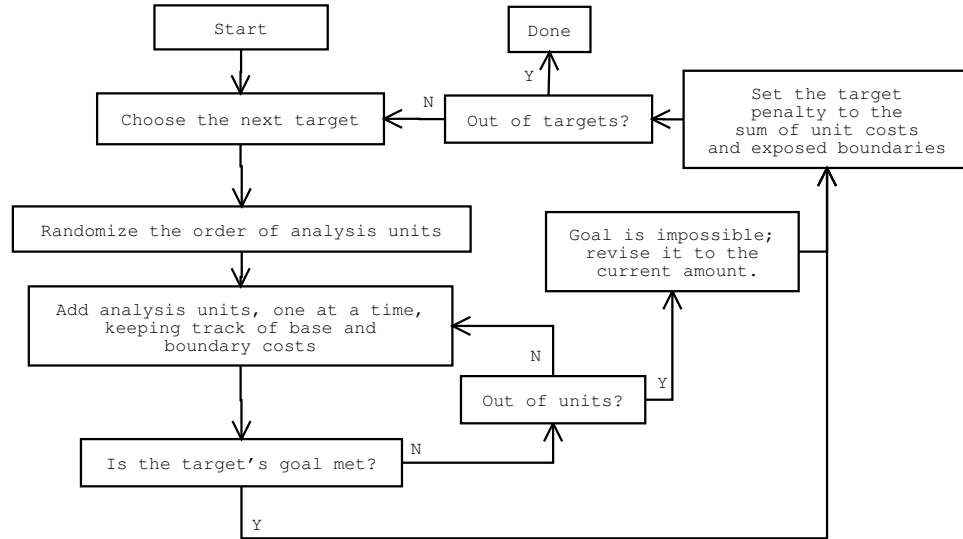


Figure B.2: Calculating the target penalties at the beginning of a run.

B.1.3 Generating a new portfolio

To generate a new portfolio, SPOT changes the status of a single non-locked analysis unit. In the future, SPOT may make larger changes depending on the temperature.

B.1.4 Evaluating the new cost

The new portfolio is run through the cost function to come up with a total cost. The new cost is compared with the current cost to see if it should replace the portfolio.

If the new portfolio is an improvement, it is accepted. Otherwise, SPOT may still accept the portfolio, even though the cost is worse, depending on the temperature and a random coin-flip.

B.1.5 The acceptance chance of a portfolio

The chance c that SPOT will accept a portfolio that has a chg higher cost is:

$$c = e^{(-chg/temperature)} \quad (\text{B.1})$$

A random coin is flipped, with a c chance of being successful. If it is, then the new portfolio is accepted; otherwise SPOT keeps the old one and moves on to the next iteration. This is logged as “FLIPCHANCE.”

B.1.6 Updating the temperature

SPOT uses an adaptive schedule to adjust the temperature and decide whether a portfolio is acceptable over the course of the run.

This schedule is known as the Modified Lam schedule, derived from the Lam annealing schedule, which uses sophisticated statistical techniques to adapt the rate of temperature decreases and ensure an adequate sampling of the cost function during a run.

The Lam schedule is complicated to implement and has the big disadvantage that runtime (number of iterations) cannot be determined in advance. However, one feature of the Lam schedule is that a much easier-to-calculate statistical metric, the ratio of accepted changes to rejected changes, generally follows the following curve over the course of a run (d is the proportion of the run that is complete):

$$TargetAcceptanceRate = \begin{cases} 0.44 + 0.56 * 560^{-d/0.15} & 0 \leq d < 0.15 \\ 0.44 & 0.15 \leq d < 0.65 \\ 0.44 * 440^{-(d-0.65)/0.35} & 0.65 \leq d < 1 \end{cases} \quad (B.2)$$

At the beginning of the run, almost all proposed portfolios are accepted, dropping down to 0.44 during the first third of the run.

The middle third of the run maintains the acceptance rate of 0.44, then during the final portion the rate drops exponentially toward zero.

This result led to the adaption of the Modified Lam schedule, which uses this rate as a target and modifies the annealing temperature to get the actual acceptance rate to match, as closely as possible, this idealized acceptance rate. SPOT outputs a log of the target rate as “TARGET_RATE.”

After each iteration, the current acceptance rate is calculated (within a 500-iteration moving average):

$$AcceptRate_i = 1/500(499 * AcceptRate_{i-1} + 1) \quad (B.3)$$

if the last portfolio was accepted, and

$$AcceptRate_i = 1/500(499 * AcceptRate_{i-1}) \quad (B.4)$$

if not.

The current acceptance rate is logged as “ACCEPT_RATE.”

After every iteration, SPOT incrementally raises or lowers the temperature depending on whether the acceptance rate is below or above the target rate.

$$temperature_{new} = \begin{cases} temperature_{old} * 0.999 & AcceptRate > TargetRate \\ temperature_{old} * (1.0/0.999) & AcceptRate \leq TargetRate \end{cases} \quad (B.5)$$

SPOT then increments its iteration count, generates a new candidate portfolio, and repeats this process until the run is complete.

Appendix C

System Requirements

C.1 Hardware Requirements

Since SPOT is very computationally intensive, the ideal machine would have a very fast processor; RAM and hard drive space are slightly less important for SPOT, but since the same machine will probably also be needed for various GIS tasks, more is better as well.

- Recommended: 1Ghz (or faster) Intel processor. Required: Pentium or newer processor.
- Recommended: 512MB (or more) RAM. Required: Varies with size of run. If there's enough RAM to run ArcView, SPOT should run.
- Recommended: 30GB (or larger) hard drive. Required: A drive with enough space to perform basic GIS tasks and store SPOT's inputs and results.

C.2 Software Requirements

- Windows 2000 or XP Professional recommended; Windows NT, Windows 98 or higher required.
- ArcView 3.x, with Spatial Analyst. SPOT was developed with ArcView 3.2, but 3.3 works well. ArcView and Spatial Analyst are not required for the SPOT slave for distributed runs.

Appendix D

Glossary

analysis unit	Analysis units form the portfolio; they are small areas that are marked as in or out to create a portfolio.
annealing	The technique of slowly cooling a liquid into a solid such that its final form is a near-optimal crystal.
base cost	A component of the SPOT cost function that encourages SPOT to minimize the area of the portfolio. It is the sum of the cost specified for each analysis unit included in the portfolio.
boundary cost	A component of the SPOT cost function, aimed toward minimizing a portfolio's fragmentation by minimizing the length of its boundary.
boundary length	The length of a portfolio's perimeter.
boundary length modifier	A multiplier that converts and scales the boundary length of a portfolio before adding it to that portfolio's cost.
core	The SPOT module that performs the annealing process to create a portfolio
cost function	The cost function for SPOT calculates a single cost value for a given portfolio that represents its effectiveness.
iteration	A single change to the current state and re-evaluation of the cost function.

local minimum	A low point in the cost function, but not the absolute lowest point.
minimum area	The minimum contiguous amount of a target required for it to contribute to a conservation goal.
penalty factor	Sets the importance of representing a target, relative to other targets and the base and boundary costs
shortfall cost	A component of the SPOT cost function that penalizes portfolios that don't meet conservation goals.
simulated annealing	A general technique for finding the lowest value of a function through many trial runs and repeated adjustment to input values.
target	A biological feature with a conservation goal that SPOT attempts to meet during assembly.
target patch	An occurrence of a target that can spread over several neighboring analysis units.
temperature	In simulated annealing, this defines the maximum allowable change in the cost function

Bibliography

- [1] Sandy Andelman, Ian Ball, Frank Davis, and David Stoms. *SITES v. 1.0: An Analytical Toolbox for Designing Ecoregional Conservation Portfolios*, December 1999.
- [2] Ian Ball and Hugh Possingham. *Marxan v. 1.7: Marine Reserve Design using Spatially Explicit Annealing*, March 2000.
- [3] Justin Andrew Boyan. *Learning Evaluation Functions for Global Optimization*. PhD thesis, Carnegie Mellon University, August 1998.
- [4] Richard C. Allen et al. *Mathematical Optimization*. 1996. Chapter from the Computational Science Education Project. URL: <http://csep1.phy.ohl.gov/mo/mo.html>.
- [5] Jimmy Lam and Jean-Marc Delosme. Performance of a new annealing schedule. In *Proceedings of the 25th ACM/IEEE conference on Design automation*, pages 306–311. IEEE Computer Society Press, 1988.
- [6] Mark D. McDonnell, Hugh P. Possingham, Ian R. Ball, and Elizabeth A. Cousins. Mathematical methods for spatially cohesive reserve design. *Environmental Modeling and Assessment*, (7):107–114, 2002.
- [7] Hugh Possingham, Ian Ball, and Sandy Andelman. Mathematical methods for identifying representative reserve networks. In S. Ferson and M. Burgman, editors, *Quantative Methods for Conservation Biology*, chapter 17, pages 291–305. Springer-Verlag, 2000.