

Dual-scale Landscape for LANDIS-II Concepts & Design

James B. Domingo
Robert M. Scheller
Catherine Ravenscroft

University of Wisconsin-Madison

Last Revised: July 24, 2006

Table of Contents

1	INTRODUCTION.....	3
1.1	References.....	3
1.2	Acknowledgements.....	3
2	DUAL-SCALE LANDSCAPE.....	4
2.1	Motivation.....	4
2.2	Relationship Between Scales	4
2.3	Site Locations	5
2.4	A Site’s Neighbors.....	6
3	DESIGN CRITERIA.....	8
3.1	Dual-scale: Replacement or Alternative?	8
3.2	Compatibility with Existing Broad-scale Extensions.....	8
3.2.1	<i>Broad-scale Access to Site Variables.....</i>	8
4	INPUT DATA FOR DUAL-SCALE.....	10
4.1	Dual-Scale in Scenarios	10
4.1.1	<i>New DualScale Parameter.....</i>	10
4.1.2	<i>Broad-scale Input Maps.....</i>	11
4.2	Input Maps for Extensions	11
5	DESIGN MODIFICATIONS TO ALGORITHMS.....	14
5.1	Succession Component	14
5.1.1	<i>Seeding.....</i>	14
5.2	Disturbances	14
5.3	Wind Extension.....	14
5.3.1	<i>Event Spreading.....</i>	14
5.4	Harvest Extension	14
6	DESIGN MODIFICATIONS TO LANDSCAPE MODULE	15
6.1	Sites	15
6.1.1	<i>Scale Property.....</i>	15
6.1.2	<i>IsActive Property.....</i>	15
6.1.3	<i>IsSubdivided Property.....</i>	15
6.1.4	<i>Area Property.....</i>	15
6.1.5	<i>Parent Property.....</i>	15
6.1.6	<i>GetEnumerator Method</i>	15
6.1.7	<i>AllChildren Property.....</i>	16
6.1.8	<i>GetChild Method.....</i>	16
6.1.9	<i>IsMutable Property</i>	16
6.2	Site Variables.....	17
6.2.1	<i>Large Subdivided Sites.....</i>	17
6.3	Landscape	18
6.3.1	<i>Active Site Indexer</i>	18
6.3.2	<i>GetSite Method.....</i>	18
6.3.3	<i>IsValid Method.....</i>	19

6.3.4 *Default Site Iterator* 19
6.3.5 *AllSites Property* 19
6.3.6 *AllLargeSites Property*..... 19
6.3.7 *ActiveLargeSites Property* 19

1 Introduction

The purpose of this document is to describe the concepts and design of dual-scale landscapes for the LANDIS-II model. The primary audience for this document is the team which will be developing this functionality for LANDIS-II. Since that team includes ecologists and developers, this document must be understandable by both types of team members. The document assumes the reader is familiar with the conceptual description of the LANDIS-II model.

1.1 References

LANDIS-II Model v5.1 Conceptual Description.

LANDIS-II Model User Guide.

TNC proposal?

1.2 Acknowledgements

Funding for this work was provided by ...

2 Dual-scale Landscape

2.1 Motivation

Currently, the landscape in the LANDIS-II model is represented by a 2-dimensional grid of equally-sized squares called **sites** or **cells** (see chapter 3 in the *LANDIS-II Conceptual Model Description*).

The motivation for the dual-scale landscape is to allow the user to subdivide selected areas of the landscape into smaller sites (see Figure 1). This functionality would allow the user to investigate the model's behavior in response to spatial variability in topography, soils, climate, etc. at a finer scale in selected regions of the landscape.

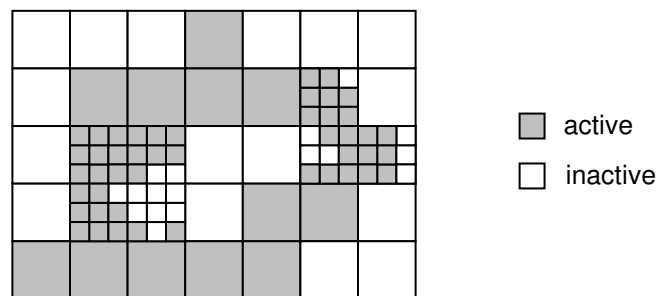


Figure 1 – Dual-scale landscape: selected sites are subdivided.

Without this functionality, the user has to model the whole landscape at the finer scale, which requires significantly more computational resources than modeling the landscape at the broader scale. For example, simulating a landscape at the smaller resolution of 28.5 m (0.08 ha per site) requires approximately 36 times the memory and processor time than simulating it at the larger resolution of 171 m (2.9 ha per site).

2.2 Relationship Between Scales

Like the larger sites, the smaller sites are equally-sized squares. Furthermore, a large site is subdivided into complete small sites (i.e., there are no partial small sites). Therefore, the cell length of a large site is an integer multiple of the cell length of a small site:

Cell length $_{large-scale} = k \times$ Cell length $_{fine-scale}$

where k is integer ≥ 2

Allow $k = 1$? Would be single-scale essentially.

2.3 Site Locations

In the current single-scale landscape, a site is identified by its location (row, column). In a dual-scale landscape, a site's location is expressed in terms of rows and columns that are based on the site's scale. So, a large site's location is expressed in terms of large-scale rows and columns, while a small site's location is expressed in terms of small-scale rows and columns.

Consider the example dual-scale landscape in Figure 2. The location of the large subdivided site F is (2, 2) while the location of the small site F7 is (6, 4).

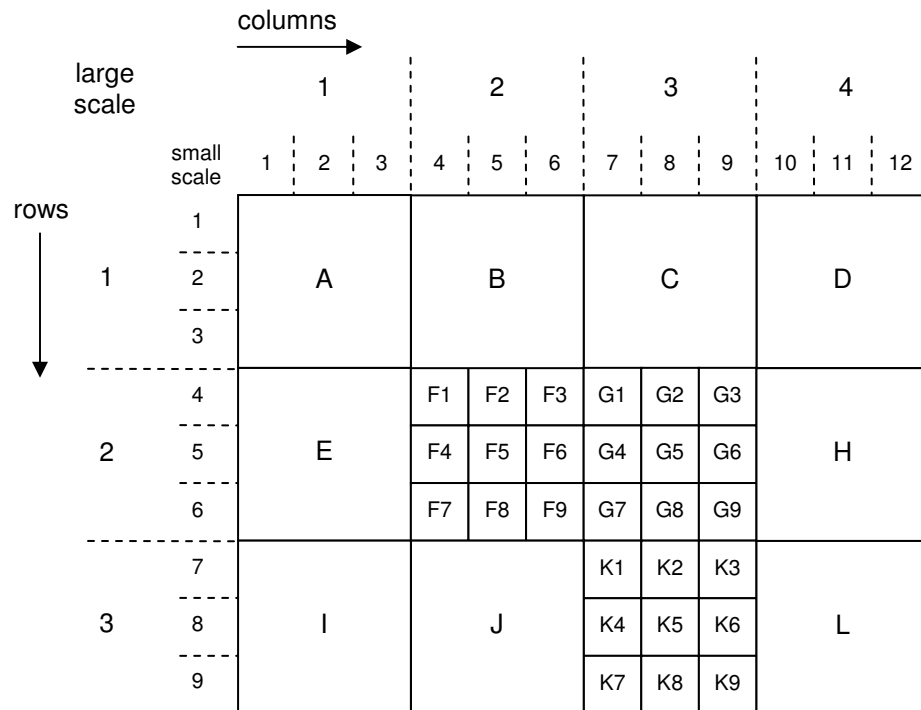


Figure 2 – Rows and columns at two scales

As a consequence of adding a second scale to the landscape, a site cannot be identified solely by its location. On a dual-scale landscape, a location and a scale are required to identify or retrieve a site. For example, in Figure 2, retrieving the site at “(3, 2), large scale” yields site J, while retrieving the site at “(5, 9), small scale” yields site G6.

Note: retrieving a site with a small-scale location may result in a large site. For example, in Figure 2, retrieving the site at “(1,4), small scale” yields the large site B.

2.4 A Site’s Neighbors

The neighbors of a site are specified by their relative location to the site. A relative location is a pair of offsets (row, column). Each offset can be positive, zero, or negative.

In a dual-scale landscape, the offsets in a relative location are interpreted in terms of the site's scale. So a relative location of a large site's neighbor is interpreted in terms of large-scale rows and columns. Similarly, a relative location of a small site's neighbor is interpreted in terms of small-scale rows and columns.

Note: When a neighbor is retrieved for a large site, the neighbor is always a large site. When a neighbor is retrieved for a small site, the neighbor may be either another small site or a large site.

Here are some neighbors for some of the sites in the example dual-scale landscape in Figure 2.

<u>Site</u>	<u>Relative Location</u>	<u>Neighbor</u>
E	(-1 , 2)	C
E	(0 , 2)	G
E	(1 , 2)	K
F9	(1 , 0)	J
F9	(1 , 1)	K1
F9	(1 , 2)	K2
F9	(1 , 3)	K3
F9	(1 , 4)	L
F9	(1 , 5)	L
F9	(1 , 6)	L

3 Design Criteria

3.1 Dual-scale: Replacement or Alternative?

Will dual-scale (DS) LANDIS-II be a separate version of LANDIS-II, available to the public (the whole LANDIS-II user community) along with single-scale (SS) LANDIS-II? Or will DS LANDIS-II eventually replace SS LANDIS-II, becoming the only version of LANDIS-II distributed?

3.2 Compatibility with Existing Single-scale Extensions

Can existing extensions that are written for SS LANDIS-II work with DS LANDIS-II? If the assumption is that these extensions are operating at the broad-scale, then the dual-scale would have to support broad-scale operations through the current landscape interface. For example, in the current interface for a single-scale landscape, the default iterator for the landscape traverses all the active sites. In order for a dual-scale landscape to support the single-scale interface, it would have to provide an iterator that traverses all the active broad-scale sites (i.e., large sites). This should be relatively straight-forward to implement.

3.2.1 Broad-scale Access to Site Variables

However, the concept of the dual-scale landscape has a significant impact on site variables (see section 6.2 for details). An existing single-scale extension would be accessing site variables at the broad scale. This raises an important question: what happens when a site variable is indexed with a large site that is subdivided?

For example, consider the current output extension that generates shade maps. This extension accesses the shade site-variable which is assigned by the succession extension. With a dual-scale landscape, the shade site-variable stores shade values for large undivided sites and for the small sites of each large subdivided site. So what happens when the output extension indexes the shade site-variable with a large site that is subdivided?

One option is to generate a single shade value for the large site from the shade values of its small sites. The broad-scale shade value may be the largest value among the fine-scale values, or it may be the average of those values. Section 6.2.1 outlines a technique to

accomplish this without any changes to the output extension's source-code.

The technique works satisfactorily for site variables that store simple data types (e.g., numbers) for which there are clearly-defined ways to generate a broad-scale value from fine-scale value. But issues arise with site variables that store more complex data types like groups of cohorts at a site. For these data types, it may be really difficult (or impossible) to define a technique for generating a broad-scale value from fine-scale values.

For example, the succession extension creates a site-variable to store a group of cohorts for each active site. With a dual-scale landscape, this site variable has a group of cohorts for each large active undivided site and for each small active site.

Now consider the existing "Max Species Age" output extension which generates maps for maximum cohort age for selected species. This extension iterates through the active broad-scale sites on the landscape, gets the group of cohorts at each site from the cohort site-variable, and passes each cohort group to a utility function that computes the maximum cohort age at the site.

In order to avoid changing this extension's source code, somehow the cohort site-variable would need return a broad-scale cohort object when the variable is indexed with a broad-scale site that's divided. Not sure if it's possible to define such a broad-scale group of cohorts from the groups of cohorts at the fine-scale sites.

4 Input and Output Data

4.1 Dual-Scale in Scenarios

- How does the user specify that a scenario uses a dual-scale landscape?
- How does the user specify the resolution of the two scales?
- How does the user indicate which large sites are subdivided?
- How does the user indicate which small sites are active?

Current idea is to have the ecoregions map at the fine scale. Then the user uses a new parameter to denote the integer multiplier for the broad scale (k in Equation 1).

4.1.1 New DualScale Parameter

This is a new parameter for the scenario file (see chapter 4 in the *LANDIS-II Model User Guide*). It comes after the CellLength parameter. The parameter is optional to maintain backward compatibility. If the parameter is not present, then the landscape is single-scale.

If the parameter is present, then the resolution of the ecoregions map is the fine scale, and the parameter's value represents the multiplier for computing the broad scale.

```

LandisData      Scenario
...
Ecoregions      path/to/ecoregions.txt
EcoregionsMap   path/to/ecoregions/map.gis

CellLength      100  << meters

DualScale       5    << broad-scale multipiler

InitialCommunities      path/to/init-communities.txt
InitialCommunitiesMap   path/to/init-communities.gis
...

```

If the parameter is present, the initial-communities map is expected to be at the same resolution of the ecoregions map (i.e., fine-scale).

*Maybe this parameter should be named `BroadScaleMultiplier?`
`BroadScaleFactor?`*

Should this parameter be located before the `Ecoregions` parameter?

4.1.2 Broad-scale Input Maps

Idea for having the ecoregions and initial-community maps at broad scale. Map codes would indicate whether a broad-scale site is divided or not and how it's divided. For example, one map code may represent that the site belongs to ecoregion X and is undivided. Another map code may represent that the site is divided with its leftmost column of small sites all belonging to ecoregion X and all the other small sites belonging to ecoregion Y.

The number of map codes needed would depend the broad-scale multiplier (k , i.e., the number of small sites per large site), and how many combinations exist on the landscape.

Need to describe this idea more fully. It'd likely be more work for the modeler to prepare these type of input maps, but they'd be smaller files. Just wanted to record the idea.

4.2 Input Maps for Extensions

An input map can either be at the large scale or the fine scale.

The resolution for an input map for an extension is based on the scale at which the extension operates. If the extension has been developed to operate at the broader scale, then it will expect its input maps to be at the larger resolution. If the extension has been developed to operate at the finer scale, then it will expect finer-resolution input maps.

A broad-scale extension is responsible for handling pixel data for large sites that are subdivided. Depending upon the type of input data, the extension may replicate the large site's data value for all its small sites, or the extension may subdivide the large site's data value among all its small sites.

Conversely, a fine-scale extension is responsible for handling pixel data for large sites that are not subdivided. For a large undivided site, the extension must combine the pixel data for all the corresponding small sites in the input map into a single data value. Depending upon

the type of input data, this combination may be a summation or an average.

4.3 Output Maps

How does an extension generate an output map at fine-scale? Currently, to generate a single-scale output map, the extension iterates through all the sites (active and inactive) on the landscape, so it can generate pixel values for the output map.

```
// Open the output map for writing
foreach (Site site in Model.Core.Landscape.AllSites)
{
    // compute a pixel value for the site
    // write the pixel value to the map
}
```

What would the loop for a fine-scale map look like? Specifically, what sort of iterator do we need for the landscape? It would appear that we would need an iterator that goes through all the fine-scale sites on the landscape. Problem: the landscape is not divided into fine-scale sites everywhere.

Consider the sample dual-scale landscape in Figure 2. In order to generate pixel values for a fine-scale output map, we need to access the large undivided sites multiple times:

<u>Pixel Location</u>	<u>Site</u>
(1 , 1)	A
(1 , 2)	A
(1 , 3)	A
(1 , 4)	B
(1 , 5)	B
(1 , 6)	B
...	...
(1 , 11)	D
(1 , 12)	D

This same sequence of sites would be repeated for the pixel locations in rows 2 and 3. For row 4, this is the sequence of sites accessed:

<u>Pixel Location</u>	<u>Site</u>
(4 , 1)	E
(4 , 2)	E
(4 , 3)	E
(4 , 4)	F1
(4 , 5)	F2
(4 , 6)	F3
(4 , 7)	G1
(4 , 8)	G2
(4 , 9)	G3
(4 , 10)	H
(4 , 11)	H
(4 , 12)	H

One approach: we create an iterator that returns all sites in a landscape at fine scale, recognizing that we will get back both large and small scale sites.

Second approach: we create an iterator for all the site locations on a landscape at fine scale, and then use the landscape's GetSite(location, "Fine scale") method to get each site in turn.

Drawbacks of 2nd approach is that inefficiency. Since we're explicitly getting each site, we can't construct an enumerator that remembers where it is in the landscape, and hence retrieving the next site would be quicker. But is this really a significant loss?

5 Design Modifications to Algorithms

5.1 Succession Component

- Sufficient Light – Operates at the fine scale because the shade at a site is based on the cohorts present, and each small site has its own set of cohorts.
- Establishment – Operates at the fine scale, since each small site is associated with a particular ecoregion.

These two conditions affect various forms of reproduction.

5.1.1 Seeding

Mature Present – At large scale: a species is present a large divided site if it's present at least one of the small sites.

5.2 Disturbances

How does dual-scale impact the disturbance interfaces in the cohort libraries, and the implementation of these interfaces in various disturbance extensions?

5.3 Wind Extension

5.3.1 Event Spreading

To do.

5.3.2 Severity Maps

How is site severity determined for a large divided site?

5.4 Harvest Extension

To do.

6 Design Modifications to Landscape Module

Need to analyze how various dual-scale extensions will use the landscape module in order to finalize the design of the module's modifications.

6.1 Sites

6.1.1 Scale Property

This new property indicates the site's scale. Value: Large or Small.

6.1.2 IsActive Property

This existing property indicates whether the site is active or not.
Value: true or false.

By definition, a large subdivided site is active if at least one of its small sites is active.

6.1.3 IsSubdivided Property

This new property indicates whether the site is subdivided into smaller sites. Value: true or false.

This property is always false for small sites.

6.1.4 Area Property

This new property is the area of site. Value: number > 0. Units: m².

This may be useful for extensions that need to know this. Currently, the core provides extensions with two properties: CellLength and CellArea. Perhaps these should be moved to the Landscape class?

6.1.5 Parent Property

This new property is the site's parent: the large-scale site that encompasses the site. Value: a large-scale site.

By definition, the parent of a large-scale site is itself.

Alternatives to "Parent" as a name? "EnclosingLargeSite", "MyLargeSite"

6.1.6 GetEnumerator Method

This new method will be added to allow the developer to iterate through the small active sites in a large divided site.

```
foreach (Site smallSite in largeSite) {  
    ...  
}
```

If the site is a small site, this method returns an enumerator which iterates through one site: the small site itself. If the site is a large undivided site, this method returns an “empty” enumerator which doesn’t iterate through any small sites.

6.1.7 AllChildren Property

This new property returns an enumerator that iterates through all the small sites (active and inactive) for a large subdivided site. If the site is a large undivided site, then this property returns an “empty” enumerator. If the site is a small site, then the enumerator iterates through just one site: the small site itself.

Alternative name: “AllSmallSites”

6.1.8 GetChild Method

This new method returns a small site within a large subdivided site. The parameter is the small site’s location within the large site. Therefore, if the parameter is location (1,1), then the upper-left small site is returned. If the location is out-of-bounds (row or column is not between 1 and k), then the method returns a null site.

Null site is returned by the default constructor of the new Site value type. Its location is (0, 0) (returned by the Location’s default constructor), and its landscape is null. Would a new boolean property called IsNull be useful? Alternative names: IsZero, IsValid, Exists

6.1.9 IsMutable Property

This existing property will be removed. It’s needed because sites are implemented as objects (i.e., a reference type), and site iterators (enumerators) use mutable versions of site objects to minimize object creation.

The development of various extensions has demonstrated the need for immutable versions of site objects. But according to the .NET guidelines for value types, sites should be implemented as a value type (structs in C#). Since value types are stack-based and utilize copy semantics for assignment operations, they are immutable.

Changing the Site type from a reference type to a value type would eliminate the need to implement an ImmutableSite type and would allow the elimination of existing mutable classes.

This change would break binary compatibility for 5.1 extensions. However, source compatibility may still be possible (source compatibility = no changes to source code, just need to recompile the extension).

6.2 Site Variables

Site variables are data structures for storing site-specific information. A site variable has a distinct value for each active site on the landscape. A site variable handle inactive sites in one of two ways: 1) all the inactive sites share a single common value, or 2) each inactive site has its own distinct value. The former variation uses less memory and is used much more frequently by extensions.

6.2.1 Large Subdivided Sites

How are large subdivided sites handled? A site variable doesn't allocate memory for large subdivided sites; it only allocates memory for small active sites and large active undivided sites. In other words, if a site variable is indexed with a large subdivided site, what happens?

One option is to raise an `InvalidOperationException`. With this option, any code using a site variable has to check for large subdivided sites before indexing the site variable.

Another option is to have two new delegate properties which are called whenever a large subdivided site is used with a site variable.

```
class SiteVariable
{
    static class Delegates
    {
        delegate T GetSubdividedSite<T>(Site site);
        delegate void SetSubdividedSite<T>(Site site,
                                           T value);
    }
}

interface ISiteVar<T>
: SiteVariable
{
    Delegates.GetSubdividedSite<T> GetSubdividedSite
    { get; set; }
```

```
        Delegates.SetSubdividedSite<T> SetSubdividedSite
            { get; set; }
    }

    // example of use

    static class FooUtil
    {
        static Foo GetSubdividedSite(Site site)
        {
            // Maybe average or total the values for
            // the site's small sites.
        }

        static void SetSubdividedSite(Site site,
                                      Foo value)
        {
            // Maybe assign the value to all the site's
            // small sites. Or, partition the value
            // among the small sites.
        }
    }

    ISiteVar<Foo> fooVar;
    fooVar = Model.Core.Landscape.NewSiteVar<Foo>();
    fooVar.GetSubdividedSite = FooUtil.GetSubdividedSite;
    fooVar.SetSubdividedSite = FooUtil.SetSubdividedSite;
```

By default, when a site variable is created on a dual-scale landscape, these two new delegate properties point to methods that simply raise an `InvalidOperationException`.

6.3 Landscape

6.3.1 Active Site Indexer

The parameter is the site's location.

Should this location be interpreted at the large scale? This would mean that existing source code would be treated as operating at the large scale.

6.3.2 GetSite Method

The parameter is the site's location.

Should this location be interpreted at the large scale? This would mean that existing source code would be treated as operating at the large scale.

Need to define an overload of this method that takes a second parameter which is a scale value. The second parameter indicates which scale to use for interpreting the location.

6.3.3 IsValid Method

This existing method determines if a location is valid for the landscape.

Should this location be interpreted at the large scale? This would mean that existing source code would be treated as operating at the large scale.

Need to define an overload of this method that takes a second parameter which is a scale value. The second parameter indicates which scale to use for interpreting the location.

6.3.4 Default Site Iterator

Currently, with a single-scale landscape, the GetEnumerator method of a landscape returns an enumerator that iterates over all the active sites in the landscape.

With a dual-scale landscape, should this enumerator iterate over all active sites (large and small)? Or should it iterate over just the large active sites? The latter approach would mean that existing source code would be treated as iterating over active sites at the broad scale.

6.3.5 AllSites Property

This existing property returns an enumerator that iterates through all the sites (active and inactive) on a landscape.

For a dual-scale landscape, should this enumerator iterate over all sites (large and small)? Or should it iterate over just the large sites (active and inactive)?

6.3.6 AllLargeSites Property

This new property would return an enumerator that iterates through all the large sites on a landscape.

6.3.7 ActiveLargeSites Property

This new property would return an enumerator that iterates through all the active large sites on a landscape.