1 Quick Start

This synthetic generator can be used to study landscape connectivity conservation planning, considering both economic and ecological aspects, as well as multiple species. The generator creates randomized problem instances that represent a landscape associated with land costs and several species. For each species on the landscape, the generator creates a file describing the core habitat areas and a file describing the species-specific resistance of each cell in the landscape. The generator release includes source code and a pre-packaged benchmark of 500 instances, and can be found at `http://www.cis.cornell.edu/ics/projects/index.php`.

For any questions regarding on the generator, please contact Yexiang Xue at *yexiang@cs.cornell.edu*, Bistra Dilkina at *bistra@cs.cornell.edu* or Carla Gomes at *gomes@cs.cornell.edu*.

This generator was released as part of a dataset paper in the AAAI 2013 CompSust Track. Please acknowledge the following citation when using the generator or the synthetic benchmark that comes with it:

Bistra Dilkina, Carla P. Gomes, Katherine Lai, Ronan Le Bras, Kevin S. McKelvey, Ashish Sabharwa, Michael K. Schwartz, Jordan Suter, Yexiang Xue. (2013) *Large Landscape Conservation — Synthetic and Real-world Datasets.* In AAAI.

This generator and in particular the benchmark that comes with it was used in the following paper that contains benchmark results in the context of one particular optimization setting, the *Minimal Delay Generalized Steiner Network Problem*, capturing budget-constrained planning for robust connectivity:

Ronan LeBras, Bistra Dilkina, Yexiang Xue, Carla Gomes, Kevin S. McKelvey, Michael K. Schwartz, Claire A. Montgomery (2013) *Robust Network Design for Multispecies Conservation.* In AAAI.

This package contains a general description (this file); the source code of the synthetic generator, makefile, and sample configuration files (in src/ folder) and well as illustrative sample generated instances (in examples/ folder).

The output of the genartor is a set of ascii files descrbing the landscape in *ASCII grid format*, which is file format designed to encode raster image data files. The data is presented as a martix of numbers corresponding to each cell in the raster, preceded by a preamble of 6 lines that specifies important metadata. More details on this file format can be found at `http://en.wikipedia.org/wiki/Esri_grid`. ASCII Grid format is an ascii format so it is both human readable and hardware independent; it is widely supported and easy to import and export from most GIS software.

We assume the landscape forms a N-by-N grid, also referrred to as a raster of cells, in our artificial setting. One problem instance consists of the following elements:

- **Cost Matrix**. A N-by-N grid, where each cell contains the cost to purchase the land in this cell. We assume land in core areas is free. The file name is usually **_cost.asc and is in ASCII grid format.
- **Resistance Matrix**. A N-by-N grid, one matrix per species. The value of each cell in this matrix represents the resistance of this cell of landscape to the particular species. The file name is usually **_res_'species no.'.asc and is in ASCII grid format.

  The simplest way to understand resistance values is to view them as the cost for animals to migrate through a particular land patch or the difficulty of movement through that patch. Often the connectivity that a path between two locations provides is measured as the sum of the resistances of the landscape cells on the path.
- **Terminal Matrix**. A N-by-N grid, one matrix per species. A cell marked with $0, 1, 2, 3, \ldots$ are core habitat areas for that species. -9999 is used to mark non-core areas. The file name is usually **_term_'species no.'.asc and is in ASCII grid format.
- **Terminal-Pair File**. One file per species. The first line is n, the number of core area pairs with connectivity requirements we would like to specify. It is followed by n lines, where each line contains two integers, a_i and b_i, meaning we would like to ensure a connectivity requirement between core area a_i and b_i for the particular species. The file name is usually **_termpair_'species no.'.txt. These files are optional. Different studies using the synthetically generated landscapes can consider different connectivity questions and requirements.

To use the synthetic generator, one needs to specify a configuration file first. Here is the description of a configuration file.

```
FilePrefix (string)  /* the prefix of the file name of the
                      * instance produced.
                      * ie, cost matrix file will be named as
                      *      FilePrefix_{no.}_cost.asc
                      */
STRUCTURED (string)  /* either RANDOM or STRUCTURED.
                      * specifies the approach that applies.
                      * will be explained later.
                      */
N (int)              /* generate N-by-N matrices. */
S (int)              /* split the matrix into S-by-S block
                      * matrices. It will be explained later.
                      */
m (int)              /* the number of cells each core
                      * area contains.
```

```
                      */
nSpecies (int) k (int)/* nSpecies is the number of different
                      * species and each species has
                      * 2*k core areas.
                      */
nCopy (int)          /* Create nCopy instances of this type. */
nExtra (int)         /* Parameter needed ONLY when it is in
                      * STRUCTURED mode. Specifies the number
                      * of extra Gaussian functions for
                      * each species serving as ‘‘connectors’’.
                      * will be explained later. */
sigma_center (double)/* Parameter needed ONLY when it is in
                      * STRUCTURED mode.
                      * Specifies the width of the Gaussian
                      * functions. will be explained later.
                      */
```

A configuration file can be used to create a benchmark of nCopy instances of type either RANDOM or STRUCTURED for landscapes of size N-by-N grid, for S different species with 2k core areas per species.

Sample configuration files are random_input.txt and structured_input.txt in src/ folder. After specifying the parameters, you can compile and run the synthetic generator by

```
make
```

and

```
./synthetic_gen random_input.txt
```

inside src/ folder. You could find the results in examples/ folder. displayMatrix.py helps you visualize matrices. Its usage is

```
./displayMatrix.py asc_matrix_file_name
```

It requires python, scipy and numpy installed.

We will briefly discuss how STRUCTURED and RANDOM approaches work in the following sections.

2 Design Details

2.1 Generate Terminal Matrix

We start by splitting the NxN grid of cells into SxS grid of blocks (each block is a $\lfloor N/S \rfloor$-by-$\lfloor N/S \rfloor$ square). The *outskirt* blocks are defined as those in the out-most laterals of the square. For each species i to generate the $2k$ core areas, we pick outskirt blocks (x_1, y_1), (x_2, y_2), ..., (x_{2k-1}, y_{2k-1}), (x_{2k}, y_{2k}).

The default Terminal-Pair File generated for each species has connectivity requirements for k pairs of core areas. We enforce connectivity requirements between the pairs of core areas $(2t-1, 2t)$ for all $t = 1..k$ located in blocks (x_{2t-1}, y_{2t-1}) and (x_{2t}, y_{2t}). In fact, for each $t = 1..k$, we randomly pick a block (x_{2t-1}, y_{2t-1}) among upper outskirt blocks. Once it is picked, (x_{2t}, y_{2t}) is automatically fixed at $(S-x_{2t-1}, S-y_{2t-1})$ in order to make their location of core areas $2t-1$ and t as far as possible. The blocks $(x_1, y_1), (x_3, y_3), \ldots, (x_{2k-1}, y_{2k-1})$ are sampled multiple times until they do not overlap.

Finally, for each core area $t = 1..2k$ within the chosen block (x_t, y_t), we grow a blob of size m using breath-first search to select that actual grid cells in the NxN matrix that belong to the core area.

N, S, k and m are specified in the configuration file.

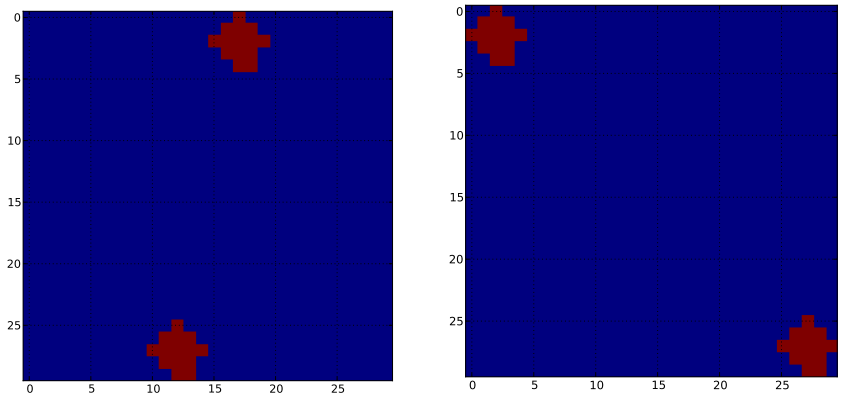

Figure 1: Terminal Matrices for $N = 30, S = 6, k = 1, m = 14$. Dark red squares are part of the terminals. (Left) Species 0 (Right) Species 1

2.2 Generate Resistance Matrix

2.2.1 Totally Randomized Approach

This approach is called when specifying RANDOM in configuration file. In this case, a matrix of size N-by-N is generated for each species, where each cell

is a uniformly distributed random number between $[0, ResistanceRange - 1]$ representing the *resistance* value of the landscape. $ResistanceRange = 2000$ in the code (but can easily be changed).

2.2.2 Structured Approach

This approach is called when specifying STRUCTURED in configuration file. In this approach, resistance matrices are created as a mixture of 2-dimensional Gaussian functions. A 2-dimensional Gaussian function $g(x, y)$ is characterized by a 5-tuple, $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ and is computed as:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\{-\frac{1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\frac{\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right)\}$$

Below we will report our choice of these five parameters without referring to the function form again. A negative Gaussian function is simply $g'(x, y) = -g(x, y)$ where $g(x, y)$ is a Gaussian function. In our case, x and y will be the row and column coordinates of each grid cell within the NxN matrix.

First, we assume low resistance values within core areas. Therefore, for each species $i = 1, 2, \ldots, nSpecies$ and core area $t = 1, 2, \ldots, 2k$ (k is the number of pairs of core areas), we put a negative Gaussian function $-G_i^t$ with a mean vector (μ_x, μ_y) corresponding to the coordinates of the matrix cell at the center of the core area, with $\sigma_x = \sigma_y = \sigma_{center}$ and $\rho = 0$ for each core area.

After that, for each species we put another $nExtra$ negative Gaussian functions with mean vectors randomly sampled from non-outskirt blocks, and with σ_x and σ_y randomly chosen in the interval $[\sigma_{center}, 1.5\sigma_{center}]$; ρ randomly chosen in the interval $[-0.8, 0.8]$. We call these functions $-\hat{G}_i^t$, where $i = 1, 2, \ldots, nSpecies$ and $t = 1, 2, \ldots, nExtra$. These functions are used to create "low resistance" areas outside core areas to allow animals to migrate between habitats. $nExtra$ and σ_{center} are parameters specified in the configuration file.

For each species i we compute a Gaussian mixture G_i, where $\hat{G}_i^t$ are combined with G_i^t with halved intensity, and the value of each cell is further perturbed with uniformly distributed white noise at the intensity of $\frac{1}{10}$ of G_i^t. In summary, the combined matrix for species i is given by:

$$G_i(x,y) = -\sum_{t=1}^{2k} \frac{G_i^t(x,y)}{max(G_i^t)} - \frac{1}{2}\sum_{t=1}^{nExtra} \frac{\hat{G}_i^t(x,y)}{max(\hat{G}_i^t)} - \frac{1}{10}U(0,1)$$

where $max(G_i^t)$ is the maximal value of G_i^t across all (x, y) cells (Similarly for $max(\hat{G}_i^t)$.) . This normalization serves to make each component of the sum have value at most 1. $U(0, 1)$ is a uniformly distributed random variable within $[0, 1)$.

Finally, G_i is normalized to guarantee the values are in the range $[1, ResistanceRange]$ ($ResistanceRange = 2000$) to form the resistance matrix R_i for species i:

$$R_i(x,y) = \begin{cases} ResistanceRange - 1 & G_i(x,y) = 0 \\ \lfloor (1 - G_i(x,y)) * ResistanceRange \rfloor & -1 < G_i(x,y) < 0 \\ 0 & G_i(x,y) \leq -1 \end{cases}$$

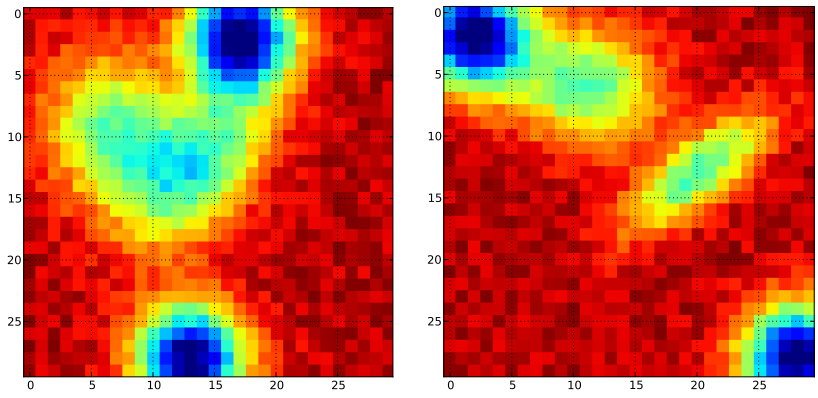

Figure 2: STRUCTURED Resistance Matrices: Blue is low resistance, red is high resistance. The figure illustrates how the areas corresponding to core areas from Fig.1 here have low resistance. The additional low resistance areas correspond to two extra randomly placed Gaussian functions for each species. $nExtra = 2, \sigma_{center} = 3$. (Left) Species 0 (Right) Species 1.

2.3 Generate Cost Matrix

2.3.1 Totally Randomized Approach

This approach is called when specifying RANDOM in configuration file. In this case, a matrix of size N-by-N is generated per problem instance, where each cell is a uniformly distributed random number between $[0, CostRange - 1]$. We also force the cost of the land at core areas to be free. The cost range $CostRange$ is set to 2000 in the code (but can easily be changed).

2.3.2 Structured Approach

This approach is called when specifying STRUCTURED in configuration file. The way structured costs are computed is motivated by the fact that costs of landscapes are usually correlated with their resistance value. High resistance landscapes often correspond to urbanized areas with high land cost, while very low resistance areas are usually undeveloped, conserved or rural areas of low land cost value.

The cost value of cell (x, y) is computed based on the minimal resistance value among the reistsance matrices for all species $R_1(x, y), R_2(x, y), \ldots, R_{nSpecies}(x, y)$, plus uniformly distributed white noise (similarly to the resistance generation). In addition, we also force the cost of the land inside core areas to be free (assuming the core areas are already under conservation). The cost is computed based on:

$$C'(x,y) = \begin{cases} 0 & (x,y) \text{ is in any core area} \\ Min_{i=1}^{nSpecies} \frac{R_i(x,y)}{ResistanceRange} + \frac{1}{10} * U(0,1) & \text{otherwise} \end{cases}$$

C' is normalized to guarantee the values are in the range $[0, CostRange - 1]$:

$$C(x,y) = \lfloor \frac{C'(x,y)}{max(C')} * CostRange \rfloor$$

Currently, $CostRange$ is 2000 in the code (but can easily be changed).

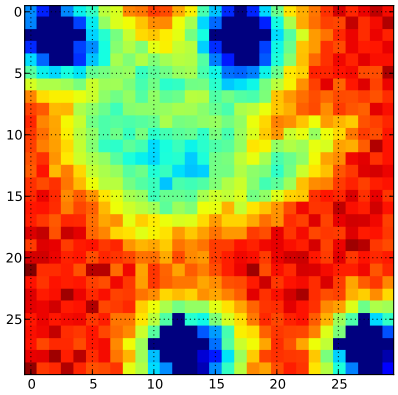

Figure 3: STRUCTURED Cost Matrix: Blue is low cost, red is high cost. In this example of a structured instance, cost is strongly correlated with landscape resistance for the 2 species from Fig.2.

3 An Example

We give a concrete example on how to generate a structured instance in this section. Firstly, we start by specifying the configuration file as shown below (src/structured_input.txt).

```
struct
STRUCTURED
30 6
```

```
14
2 1
5
2
3
```

It tells the generator that the file names should start with "struct*". We want STRUCTURED instance. The file specifies that N=30, S=6, m=14, nSpecies=2, k=1, nCopy=5, nExtra=2, and $\sigma_{center} = 3$. The instance should be a 30-by-30 matrix, divided into 6-by-6 block matrix (each block consists of 5x5 cells). Each core area contains 15 grid cells. We want 2 species, each species has 1 pair of core areas. We want 5 instances of this type. The last two parameters are used when generating structured instances specifying $nExtra$ and σ_{center} needed for the resistance generation.

After we execute the generator, for benchmark instance 0 we get files , among which struct_0_cost.asc is the cost matrix; struct_0_res_0.asc and struct_0_res_1.asc are resistance matrices for species 0 and 1, respectively; struct_0_term_0.asc and struct_0_term_1.asc are terminal matrices for the two species; and struct_0_termpair_0.txt and struct_0_termpair_1.txt are terminal pair files for the two species. Similar files are generated for the other 4 instances in the benchmark.

Using displayMatrix.py, we can display the generated .asc files. Terminal matrices and Resistance matrices are displayed in Figure 1 and Figure 2. Cost matrix is shown in Figure 3. One can notice that it is correlated with resistance matrices; however with some random perturbations. If you run the generator with the same configuration set, you might not get the exact same output as displayed in the figures, due to the randomness.