SPOMM-MGAv3 User Guide

Gary Polhill

14 February 2014

The James Hutton Institute, Craigiebuckler, Aberdeen. AB15 8QH. United Kingdom

1 Synopsis

spomm-MGAv3.pl is a Perl script designed to search for habitat and SPOMM parameters that fit an observed distribution of species and spatially-explicit biophysical data. A multicriteria genetic algorithm is used to perform this search.

SPOMM-MGAv3 requires that you have SPOMM 2.3 installed on your system, and Perl 5. (Version 5.8.8 used locally.)

Usage:

```
spomm-MGAv3.pl [-maxproc <maxproc>] [-keepall] [-
    deleteall] [-keephof] [-habmod <modifier>] [-
    fitnessnooccup] [-fitnesshalfpenalty] [-conflate <
    formula>] [-useextinct] [-usenoequilibrium] [-blob
    <blob cmd>] [-noblob] [-blobkey <blob key>] [-wd <
    dir>] [-log <file>] [-cmd <exe>] <GA file> <SPOMM
    file> <occupancy file> <biophysical data files...>
```

SPOMM-MGAv3 can also continue from a previous run, in which case the usage would be:

```
spomm-MGAv3.pl continue <file> [overriding options...]
```

The options are explained in table 1.

The GA file is a text file containing configuration parameters for the genetic algorithm. The SPOMM file is the SPOMM parameter file (given to the SPOMM 2.3 executable with the -p command line argument) to use with the simulation. The species file, and if used, habitat file referred to in the SPOMM parameter file should be present in the current working directory. The occupancy file contains observed species occupancy data for each patch. The biophysical data files are files containing observed biophysical data for each patch.

Table 1: Command-line options available to spomm-MGAv3.pl

Option	Argument	Default	Description
-maxproc	*number*	`1`	Maximum number of concurrent processes to run
-keepall		*No*	Keep data from all runs made (will use a lot of data)
-deleteall		*No*	Delete all data from all runs
-keephof		*Yes*	Keep only data from the runs saved to the 'hall of fame' – the runs throughout the search having the highest fitness
-habmod	`pct`, `none` or `sigmoid`	`pct`	Modify the habitat computed from the $\overrightarrow{\beta}$ vector (h) using the stated method. `pct` divides by 100, `sigmoid` puts the result through $1/1+e^{-h}$, `none` does nothing. Note that `sigmoid` is the default for `searchSpline*BETA`; in this case, values of the `-habmod` argument other than `sigmoid` or `none` cause an exception.
-fitnessnooccup		*No*	The fitness function should not include a comparison of total occupancy
-fitnesshalfpenalty		*No*	Add a dimension to the fitness function penalising runs generating lots of cells with 0.5 occupancy probability.
-conflate	`no`, `sum`, `product`, `wsum(`*weights...*`)` or `wproduct(`*weights...*`)`	*No*	Use a single criterion fitness function by converting the fitness vector to a single value. The `sum` option adds all elements of the fitness vector together to produce a scalar fitness value. The `product` option multiplies them. The `wsum` option allows you to compute a weighted sum: the *weights...* argument is a comma-separated (no spaces) list of weights to use for each element of the fitness vector (see section 3). The `wproduct` option allows you to compute a weighted product. Here the *weights...* argument is a comma-separated list of powers to raise each element of the fitness vector to before computing the product.
-useextinct		*No*	Runs involving total extinction should compare occupancy with the observed data
-usenoequilibrium		*No*	Do not treat runs that fail to reach equilibrium as automatically inferior to those that do reach equilibrium
-blob	*executable R script*	Irrelevant unless you are using this at MLURI	Location of R script to produce 'blob' plots. See Section 4.
-noblob		*No*	Do not produce 'blob' plots.
-blobkey	*integer*	20	Number of levels in 'blob' plots.
-wd	*directory*	`/var/tmp/spomm-GA/`*pid*	Working directory to save results of the simulation
-log	*file*	`LOG`	Log file. If this does not begin with /, it will be stored in the working directory
-cmd	*command*	Irrelevant unless you are using this at MLURI	SPOMM executable to run

The GA is expected to find SPOMM parameters specified by the user and a vector $\overrightarrow{\beta}$ with one element for each biophysical file such that for each patch i, $h_i = \overrightarrow{\beta}.\overrightarrow{b_i}$, where h_i is the habitat on patch i ($0 \leqslant h_i \leqslant 1$), and $\overrightarrow{b_i}$ is the vector of biophysical characteristics, to maximise the fit to the observed occupancy.

When using the `continue` command, the next argument should be the name of a file saved by SPOMM-MGA in the working directory (with name state-*pid*.txt). This file contains information on the working directory and current working directory[1] of the previous run, its command-line options, population and fitnesses. In continuation mode, the program will reuse as much data as it can. It expects the GA file, SPOMM files and biophysical data files used in the previous run to be in the same place as they were then, and to have been unchanged. (Some changes to the GA file might work, but not to population size, beta names or SPOMM parameters searched.) It will also re-use any command-line options you gave to the previous run, though you can override some of them by giving them after the continuation file argument. If the run data files are still available in the original working directory, they will be re-used, otherwise fitnesses for each member of the population will be recomputed.

Note that the program, depending on its configuration, could take a very long time to run, and, particularly if the -keepall command-line option is given, use a great deal of disk space.

2 File formats

2.1 GA file

The GA file is formatted thus:

```
population <population>
generations <generations>
rule <rule>
parameters {
        <param name> = <param value>
        ...
}
searchSPOMM {
        <SPOMM file> / <SPOMM parameter> : <prior> <constraint>
        ...
}
searchBETA|searchGaussianBETA|searchPoly<n>BETA {
        <name[,name]|intercept|*>[^{M|S|<m>}] : <prior> <constraint>
        ...
}
```

[1] Sorry for the confusion: the *working directory* of a run is that given by the -wd command-line option or its default value: the place where the results are saved. The *current working directory* of a run is the directory you were in when you executed the spomm-MGA.pl command (on Unix machines, the result of the pwd command).

`<population>` should be replaced with the size of population you want the GA to use, and `<generations>` by the number of generations you want the GA to run for. The `<rule>` should be one of `DEMC`, `DE`, `GA`, `DEMCStrict`, `DEMCRlen`, `DEMCStrictNormal`, `DEMCRlenNormal`, `DEMCZStrict`, `DEMCZRlen`, `DEMCZStrictNormal`, `DEMCZRlenNormal` or `DEStrict`. The parameters then list the parameters to be used for the rule you have selected. These are explained in table 2. The rules themselves are explained below:

`DE` Differential Evolution with arbitrary selection method (see `select` parameter in table 2). This relies on an unbounded real search space, and there are a number of implementation variants. That used here is based on the characterisation in [tB04], which generates a new proposal for a population member $\overrightarrow{g_p}$ from three randomly selected members of the population $\overrightarrow{g_a}$, $\overrightarrow{g_b}$ and $\overrightarrow{g_c}$ thus: $\overrightarrow{g_p} = \overrightarrow{g_a} + \gamma\left(\overrightarrow{g_b} - \overrightarrow{g_c}\right)$.

`DEStrict` Differential Evolution Markov Chain exactly as it appears in [tB04]. This uses the `posreplacebetter` selection method, which replaces $\overrightarrow{g_a}$ with $\overrightarrow{g_p}$ if $f\left(\overrightarrow{g_p}\right) > f\left(\overrightarrow{g_a}\right)$, where $f(g)$ is the fitness of a genome, computed as per equation 1. For multicriteria fitness functions, this condition is implemented as $f\left(\overrightarrow{g_p}\right) \succ f\left(\overrightarrow{g_a}\right)$, where $\succ$ represents a partial order relation on two fitness vectors $\overrightarrow{\phi}$ and $\overrightarrow{\psi}$ defined by $\overrightarrow{\phi} \succ \overrightarrow{\psi} \iff \forall i : \phi_i \geqslant \psi_i$. This is a stricter requirement, causing a proposed genome that is incomparable with the existing one to be rejected. Use of this selection method may thus be better in conjunction with the -conflate option.

`DEMC` Differential Evolution Markov Chain with arbitrary selection method (see `select` parameter in table 2). This uses [tB04]'s method for generating a new proposal $\overrightarrow{g_p}$ for the ith member of the population $\overrightarrow{g_i}$ and two other randomly selected members $\overrightarrow{g_a}$ and $\overrightarrow{g_b}$ thus: $\overrightarrow{g_p} = \overrightarrow{g_i} + \gamma\left(\overrightarrow{g_a} - \overrightarrow{g_b}\right) + \overrightarrow{e}$, where $\overrightarrow{e}$ represents a small perturbation, each element of which is taken from a uniform distribution in the range [`-b`, `b`].

`DEMCStrict` Differential Evolution Markov Chain (almost) exactly as it appears in [tB04]. Modifications have had to be made to cope with multiple criteria. The `posprobreplacebetter` selection method replaces an existing member of the population g_1 with a new member g_2 with probability $\min\left(1, \overline{f_i\left(g_2\right) / f_i\left(g_1\right)}\right)$, where $f(g)$ is the fitness of the genome, computed as per equation 1, and i iterates over the elements of that vector. That is to say, the probability of the new member replacing an existing member, even if it has poorer fitnesses is given by the mean ratio of their fitnesses. Note that no check has been made that this generalisation to the multicriteria case still enables the assumptions in [tB04] to apply; use of DEMCStrict may thus be better in conjunction with the -conflate option.

`DEMCRlen` Differential Evolution Markov Chain using a different generalisation of the selection method (`lenprobreplacebetter`) than `DEMCStrict`. This computes the Metropolis ratio $r = \sqrt{\sum_i \left(f_i\left(g_2\right) / f_i\left(g_1\right)\right)^2}$, replacing g_1

Table 2: Parameters for rules in the GA

Parameter	Applicable rules	Explanation
`select`	`DEMC`, `DE`, `GA`	Selection method to use for generating the new population. One of `posreplacebetter` (see `DEStrict`), `posprobpreplacebetter` (see `DEMCStrict`), `lenprobreplacebetter` (see `DEMCRLen`), `lottery`, `top` or `new` (see `GA`).
`gamma`	`DEMC*`, `DEStrict`, any where `select = posreplacebetter` or `posprobreplacebetter`	Differential evolution parameter
`b`	`DEMC*`, any where `select = posprobreplacebetter` or `lenprobreplacebetter`, `GA`	Positive and negative bound of uniform distribution from which to sample perturbations (small relative to expected values). For the `GA` rule, this is optional, and applied after crossover and mutate
`CR`	`DEMC*`, any where `select = posprobreplacebetter` or `lenprobreplacebetter`	Crossover probability (optional; if not given, no crossover occurs)
`chains`	`DEMCZ*`	Number of chains to explore (this is parameter N in [tBV08], and $M_0 =$ population size$-N$).
`K`	`DEMCZ*`	Number of steps to run chains before saving the exploring population to matrix $\mathbf{Z}$ (earlier chains explored).
`pcrossover`	`GA`	Crossover probability (classic style)
`pmutate`	`GA`	Mutation probability (classic style)
`pperturb`	`GA`	Perturbation probability. Probability each gene is perturbed by `b` (default 1 if `b` is specified). If this parameter is defined and `b` is not, it will have no effect.
`keep`	All	How many genomes to keep in the 'hall of fame'

with g_2 with probability $\min(1, r)$. Figures 1 and 2 compare the two methods.

`DEMCZStrict` Differential Evolution Markov Chain with resampling from earlier chains, as per DEMCZ, described in [tBV08], using the `posprobreplacebetter` generalisation of the Metropolis ratio to multidimensional fitness functions.

`DEMCZRlen` Differential Evolution Markov Chain with resamping from earlier chains, using the `lenprobreplacebetter` selection method.

`DEMCStrictNormal, DEMCRlenNormal, DEMCZStrictNormal, DEMCZRlenNormal` Perturbation is done from $N(0, b)$ as per [tB06] instead of $U(-b, b)$ as per [tB04].

`GA` Standard genetic algorithm generating new members of the population from two randomly selected parents applying crossover, mutation (and optionally perturbation) operators. The `new` selection method builds the new population only from the children, and the `top` selection method builds the new population from those in the combined proposed and existing populations with the highest fitness. The `lottery` selection method allocates lottery tickets to each member of the proposed population in ascending order of fitness, the ith member of the sorted population getting i tickets.[2] The new population is then generated by choosing lottery tickets at random. To cope with multicriteria fitness functions, sorting by fitness is achieved by successively finding the Pareto front: the set of genomes such that no other genome has better fitness (as defined above using $\succ$). (Members of this set will thus all be incomparable with each other.) Having found the Pareto front in the population as a whole, the Pareto front is found in the remaining population, and so on until all members of the population are in the (descending order) sorted list.

The `searchSPOMM` section of the file specifies which SPOMM parameters you want searched.[3] These may be in the main SPOMM parameter file, or in the species file. For the SPOMM file, write `SPOMM` before the `/`, for the species file, write `species` or `spp`. The SPOMM parameter named after the `/` should correspond to a parameter in the appropriate file. For the SPOMM file, this should be one of the parameters exactly as it is named in the SPOMM parameter file (e.g. `xparam`) for the species file, this should be the column heading of the corresponding parameter (e.g. `ALPHA`, or whatever that column is called in your species file).[4]

[2] This is a fairly standard approach to breeding in GAs, but unfortunately I don't have a reference for it.

[3] Note that only numerical parameters can be explored.

[4] Note that the parameters for only one species can currently be explored by SPOMM-MGAv3.

Figure 1: Generalisation of Metropolis ratio used by DEMCStrict and DEMCZStrict

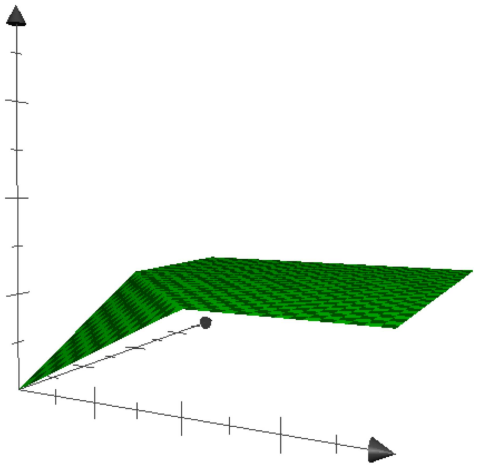


Figure 2: Generalisation of Metropolis ratio used by DEMCRlen and DEMCZRlen

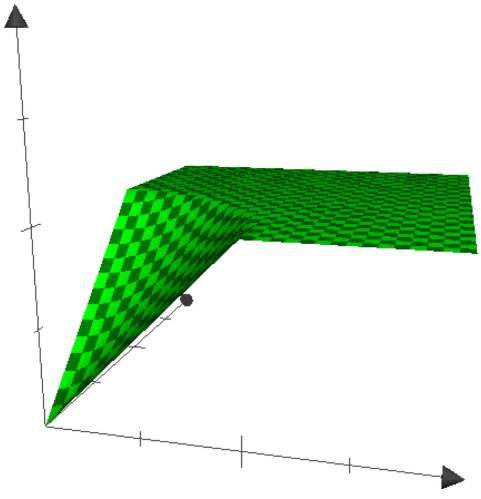


Table 3: Priors

Prior format	Description
`U(`*min*`,`*max*`)`	Uniform distribution
`N(`*mean*`,`*var*`)`	Normal distribution
`E(`*lambda*`)`	Exponential distribution

Table 4: Resampling constraints.

Constraint format	Description
`none` or `X`	No constraint
min`<X<`*max* or *min*`<=X<`*max* or *min*`<X<=`*max* or *min*`<=X<=`*max*	Range
`X<`*max* or `X<=`*max*	Upper bound
min`<X` or *min*`<=X` or `X>`*min* or `X>=`*min*	Lower bound

After the colon, the prior distributions and constraints can be specified for each SPOMM parameter. The priors specify a distribution, which may be uniform, normal or exponential. Table 3 shows the options available here. There are then two ways to impose constraints. All SPOMM parameters must be given a *resampling* constraint: a constraint that, if violated, will caused a new value to be sampled from the prior. The format and options for this are shown in table 4 The option is also available to impose constraints on the priors mathematically. Here, the sample from the prior is passed through a function before being used in the genome. This is represented in the file as a modifier on the prior, and the options for doing this are in table 5.

The `search*BETA` section of the file specifies sampling rules for each of the biophysical characteristics. The text to put before the colon depends on the mode you choose:

`searchBETA` Use `intercept`, the name of the file given on the command line to SPOMM-MGAv3 for the biophysical characteristics, a list of such names, or `*` for all names not given already. Here, the (unmodified) habitat, $h = \overrightarrow{b_i}\overrightarrow{\beta_g} + c$, where $\overrightarrow{b_i}$ is the vector of biophysical characteristics on patch i, $\overrightarrow{\beta_g}$ is the vector of linear co-efficients of genome g for these, and c is the intercept.

`searchGaussianBETA` Use the name of a file given on the command line to SPOMM-MGAv3 for the biophysical characteristics followed by a caret (`^`), followed by `M` for the mean, or `S` for the standard deviation, a list of these, or `*` for any not already given. The unmodified habitat here is $h = \prod_j \exp\left(-\left(b_{ij} - \beta_{gj}^{(\mu)}\right)^2 / \left(\beta_{gj}^{(\sigma)}\right)^2\right)$, where $\overrightarrow{\beta_g^{(\mu)}}$ is the part of the beta vector to use for the mean of the Gaussian, and $\overrightarrow{\beta_g^{(\sigma)}}$ the part to use for the standard deviation.

Table 5: Mathematically composed constraints. In the formulae, s is the sample value returned to the genome, and p is the value sampled from the *prior*, the options for which are shown in table 3.

Format	Description
`S[`*prior*`,`*min*`,`*max*`]`	Impose a range using a sigmoid: $s = min + \frac{max-min}{1+\exp(-p)}$
`T[`*prior*`,`*min*`,`*max*`]`	Impose a range using hyperbolic tangent: $s = min + \frac{1}{2} + \frac{max-min}{2}\tanh(p)$
`UV[`*prior*`,`*max*`]`	Impose an upper bound using $s = max - \|p\|$
`UE[`*prior*`,`*max*`]`	Impose an upper bound using $s = max - \exp(p)$
`Ue[`*prior*`,`*max*`]`	Impose an upper bound using $s = max - \exp(-p)$
`LV[`*prior*`,`*min*`]`	Impose a lower bound using $s = min + \|p\|$
`LE[`*prior*`,`*min*`]`	Impose a lower bound using $s = min + \exp(p)$
`Le[`*prior*`,`*min*`]`	Impose a lower bound using $s = min + \exp(-p)$

`searchPoly`***m***`BETA` Use the name of a file given on the command line to SPOMM-MGAv3 for the biophysical characteristics followed by a caret (`^`), followed by a number ($\in \{1, \ldots, m\}$), for the order of polynomial to use. Here, $h = c + \sum_j \sum_{k=1,m} b_{ij}^k \beta_{gj}^{(k)}$, using $\overrightarrow{\beta_g^{(k)}}$ to represent the part of the beta vector to use as factors of the kth power of the biophysical characteristics.

`searchSpline`***m***`BETA` Use the name of a file given on the command line to SPOMM-MGAv3 for the biophysical characteristics, followed by a caret and the letter S (`^S`), followed by a number ($\in \{1, \ldots, m\}$), for the basis matrix column number to use. Here, the biophysical characteristics file is expected to contain in columns 3, ..., $m + 2$ the basis matrix for the biophysical characteristics data in column 2 generated using the R function `ns()`. Here, $h = \left(1 + \exp\left(-\sum_j \sum_{k=1,m} s_{jk}\beta_{gj}^{(k)}\right)\right)^{-1}$, using $\overrightarrow{\beta_g^{(k)}}$ to represent the part of the beta vector g to use as the factor of the kth basis matrix column. Note that the formula for the habitat includes a sigmoid; this means the `-habmod` command line argument should not be used with this sampling rule.

After the colon, give priors and constraints, formatted as per the `searchSPOMM` section.

The following shows an example GA file:

```
population 100
generations 200
rule DEMCStrict
parameters {
        keep = 20
        gamma = 0.56
```

```
        b = 0.0001
        CR = 0.1
}
searchSPOMM {
        SPOMM / xparam : U(1,2) none
        species / ALPHA : S[N(0.9,0.1),0.3,1.3] 0.3<=X<1.3
        species / MU : E(0.5) X<=0.5
}
searchBETA {
        * : U(-10,10) none
}
```

According to [tB04], for differential evolution Markov Chain, `gamma` should be set to $2.38/\sqrt{(2d)}$, where d is the number of dimensions being searched.

2.2 Occupancy file

The occupancy file has a simple text format, in two or three columns, separated by white space, the last column of which is expected to contain the occupancy of a patch. In a two-column file, the first column should be the patch ID; in a three-column file, the first two columns should be the x and y co-ordinates of the patch. In either case, the order in which the patches appear should be exactly the same order as the SPOMM writes the patches to the `speciesPerPatchFile` (in ascending order of patch ID, sorted also by x co-ordinate first, then y).[5]

2.3 Biophysical characteristics file

The biophysical characteristics file is a comma separated value file in two columns, the first of which is the patch ID, and the second is the value of the biophysical characteristic for that patch. As mentioned above, if you use `searchSpline*BETA` for the biophysical characteristics sampling rule, then the biophysical characteristics file is expected to contain additional columns holding the basis matrix. The script `enspline.R` is available to provide the necessary conversion.

2.4 SPOMM file

Refer to the SPOMM user guide for the format of this file. Note that certain parameters in this file will get overwritten whilst the GA is executing; specifically, `speciesPerPatchFile`, `nStepListSpecies`, `patchFile` and `speciesFile`. However, you should provide a species file in the SPOMM parameter file so that the GA knows what species parameters you want to use that are not going to be searched, and the names of the parameters that will be searched. Note that you do not have to ensure that all files referred to by the `changeHabitatFile`, `autoCorrelatedFieldFile`, `goodBadYearFile`, `csvLocalizedField_file`, `landUseHabitatFile`,

[5] The easiest way to generate the occupancy file and biophysical characteristics files is to run `setup-speciesdist2.pl`.

`predationFile`, `habitatSpecificMuFile`, `sinkHabitatPropertieFile` and `occupiedPatchesPerSpecie` parameters exist in the current working directory. Those that do will be copied across; those that do not will be created as empty files.

3 Fitness function

The fitness function $f(g)$ computed by the program for each genome g is an up to seven-element vector (five if the -fitnessnoccup option is given on the command-line and the -fitnesshalfpenalty option is not given):

$$f(g) = \begin{pmatrix} p(g) \\ y(g) \\ q(g) \\ r(g) \\ m(g) \\ s(g) \\ h(g) \end{pmatrix} \tag{1}$$

The functions are defined in subsequent formulae. $p(g)$ is a fitness representing the number of patches for which the $\overrightarrow{\beta}$ vector of the genome produced a valid habitat (equation 2). $y(g)$ is a fitness representing the number of time steps the simulation ran for before total extinction (equation 3). $q(g)$ is a fitness representing the degree to which the latter half of the simulation was in equilibrium with respect to occupancy (equation 4). This is tested using a 5% two-tailed significance test comparing the distributions of occupancy in the third and fourth quarters of the simulation assuming they are normally distributed. The null hypothesis is that these two distributions are the same. $r(g)$ is a fitness representing the number of patches that have reached equilibrium occupancy (equation 5). $m(g)$ is a fitness representing the degree of match, patch by patch, between the genome's mean equilibrium occupancy and the observed occupancy (equation 6). $s(g)$ is a fitness representing the degree of similarity between the genome's total occupancy and that observed (equation 7) and is not included in the $f(g)$ vector if -fitnessnooccup is given on the command-line. $h(g)$ is a function measuring the number of patches with mean equilibrium occupancy close to 0 or 1. This is an attempt to avoid 'blobs' (see figure 3), where the search algorithm can achieve a reasonably good match by setting parameters such that the mean occupancy is close to $\frac{1}{2}$. Note that the cases in equations 6, 7 and 8 for total extinction can be switched off if the -useextinct command-line option is given.

$$p(g) = \begin{cases} 0.1 & \text{run failed} \\ 1 + \frac{\#\left\{i \in \{1,\ldots,P\} : \overrightarrow{\beta_g}.\overrightarrow{b_i} \in [0,1]\right\}}{P} & \exists i \in \{1,\ldots,P\} : \overrightarrow{\beta_g}.\overrightarrow{b_i} \notin [0,1] \\ P+2 & \text{otherwise} \end{cases} \tag{2}$$

$$y(g) = \begin{cases} 0.1 & \text{run failed} \\ 1 + t_E & \exists t_E \in \{1, ..., T\} : \Omega(t_E) = 0 \\ T + 1 & \text{otherwise} \end{cases} \tag{3}$$

$$q(g) = \begin{cases} 0.1 & \text{run failed or total extinction} \\ 1 + \frac{m}{|Z(Q_3 - Q_4)|} & |Z(Q_3 - Q_4)| > 1.96 \\ 2 + \frac{m}{1.96} & \text{otherwise} \end{cases} \tag{4}$$

$$r(g) = \begin{cases} 0.1 & \text{run failed or total extinction} \\ N_E & \text{otherwise} \end{cases} \tag{5}$$

$$m(g) = \begin{cases} 0.1 & \text{run failed or total extinction} \\ \sum_{i=1,P} 1 - \left(o_i - \overline{\omega_i(t)}\right)^2, \frac{T}{2} < t \leqslant T & \text{otherwise} \end{cases} \tag{6}$$

$$s(g) = \begin{cases} 0.1 & \text{run failed or total extinction} \\ P - \left|\sum_{i=1,P} o_i - \sum_{i=1,P} \overline{\omega_i(t)}\right| \frac{T}{2} < t \leqslant T & \text{otherwise} \end{cases} \tag{7}$$

$$h(g) = \begin{cases} 0.1 & \text{run failed or total extinction} \\ 2\sum_{i=1,P} \left|\overline{\omega_i(t)} - \frac{1}{2}\right| \frac{T}{2} < t \leqslant T & \text{otherwise} \end{cases} \tag{8}$$

Where P is the number of patches, $\overrightarrow{\beta_g}$ is the beta vector for genome g, $\overrightarrow{b_i}$ is the vector of biophysical characteristics on patch i (the dot product of these vectors being the habitat on patch i), $\Omega(t)$ is the total occupancy at time t (the number of occupied patches), T is the number of time steps the simulation is run for, t_E is the time, if it exists, at which there is zero occupancy, Q_i ($i \in \{1, 2, 3, 4\}$) is the distribution of $\Omega(t)$ for the ith quarter of the simulation (e.g. Q_1 is the distribution of $\Omega(t)$, $1 \leqslant t < \frac{T}{4}$), $Z(Q_3 - Q_4) = \left(\overline{Q_3} - \overline{Q_4}\right) / \sqrt{\sigma^2_{Q_3}/(T/4) + \sigma^2_{Q_4}/(T/4)}$ is the Z statistic of the difference in the (assumed normal) distributions of the occupancy in the third and fourth quarters of the simulation, N_E is the number of patches that have reached equilibrium, o_i is the observed occupancy on patch i, and $\omega_i(t) \in \{1, 0\}$ is the occupancy on patch i at time t.

The number of patches that have reached equilibrium is computed by conducting a binomial test, for each patch in turn, comparing the distribution of occupancy $\omega_i(t)$ in Q_3 with that in Q_4. This test uses a normal approximation to the binomial when this is expected to be good ($np \geqslant 5$ and $n(1-p) \geqslant 5$

is apparently the commonly used heuristic), but otherwise resorts to the binomial itself. Testing for similarity of two binomial distributions is achieved by computing the probability they are the same, i.e. computing P_i (using $\omega_i(Q_j)$ for the mean occupancy of patch i during the jth quarter of the simulation and $\binom{n}{r}$ for the number of combinations of n things, taken r at a time):

$$P_i = \sum_{t=1,\lfloor \frac{T}{4} \rfloor} \binom{\lfloor \frac{T}{4} \rfloor}{t}^2 \omega_i(Q_3)^t (1-\omega_i(Q_3))^{\lfloor \frac{T}{4} \rfloor - t} \omega_i(Q_4)^t (1-\omega_i(Q_4))^{\lfloor \frac{T}{4} \rfloor - t} \tag{9}$$

The fitness vector can be converted to a scalar using the -conflate command-line option, as explained in table 1. The weight vectors for the `wsum` and `wproduct` conflations should give weights for fitness components in the order given in equation 1.

4 Output

SPOMM-MGAv3 creates as output the log file containing information on progress with the run, and at the end of the simulation, the best N genomes found (N being the value of the `keep` parameter in the GA file, or the population size if this parameter is not specified) are recorded there, along with the fitness and the generation in which they were found.

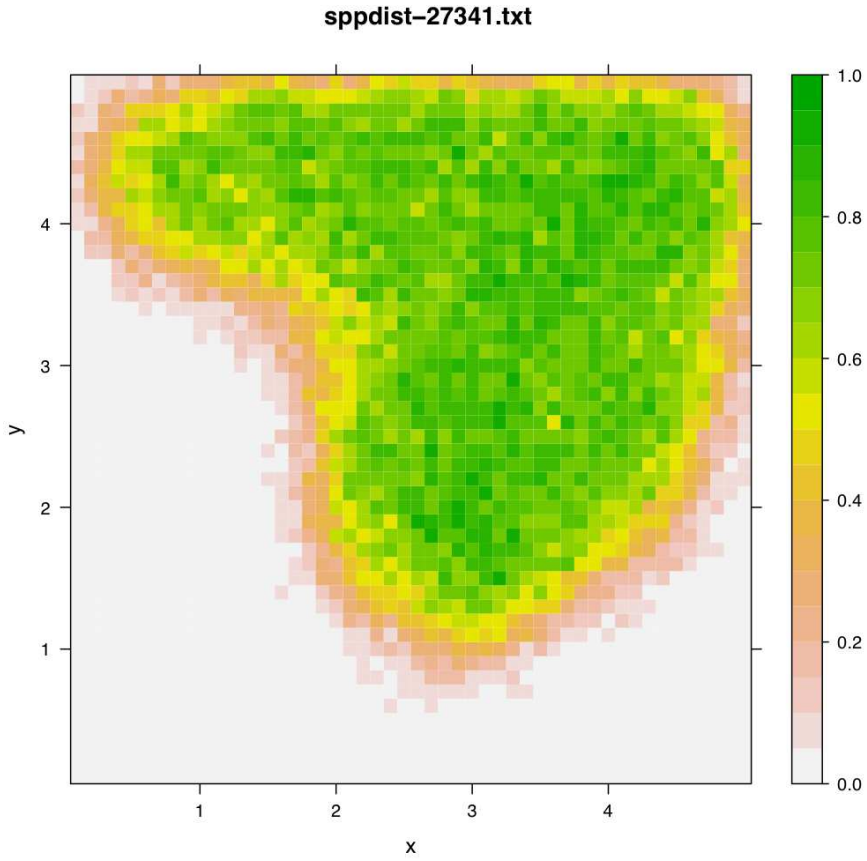
If the -keepall option is given on the command line, all files created to set up and run the SPOMM for each genome will be in a subdirectory of the working directory (either default or supplied by the argument to the -wd command line option). Each subdirectory is named by the process ID that ran the SPOMM. If GA runs for so long that process IDs are recycled, then the subdirectory named by the process ID corresponds to the most recent run with that PID; earlier runs being moved to a directory named *pid*-*n*, where n is 0 for the first use of that PID, and increases by 1 for each reuse. The log file should contain a record of which runs are stored in which subdirectories.

Otherwise, if the -deleteall option is not given, then in a directory hof will be subdirectories named as described above for the -keepall option containing the setup and output files from best N genomes found.

Included in these subdirectories is a file containing the mean species distribution over the latter half of the simulation. This will be named sppdist-*pid*.txt, and it will have the same format as the occupancy file, except that the final column will contain numbers in the range [0, 1] rather than from the set {0, 1}. If you have access to the blob.R script, and have not given -noblob on the command line, then a file named sppdist-*pid*.pdf will contain a visualisation of the distribution. Figure 3 shows some example output from this script, which is useful for inspecting whether a proposed solution is a good match for the data being modelled.

In addition, SPOMM-MGAv3 saves the population state at the final generation to the working directory in a file named state-*pid*.txt. This file is designed

Figure 3: Output from blob.R, showing why the image got its name – early results from previous versions of this program tended not to find particularly good solutions. The colour of each cell shows the probability of occupancy (over the equilibrium period), as indicated by the scale on the right hand side of the image.



to be used by subsequent runs to start from this state. However, there is no reason why you cannot create such a file yourself to store an initial population you wish to use. The format is as follows:

```
CWD
<current working directory>
<working directory>
ARGV
<command line arguments, one per line>
POPULATION
<id>: [<gene1> = <value1>, <gene2> = <value2>, ...] : (<f1>, <f2>, ...)
...
```

In the POPULATION section, there is one line per population member. The ID is a process ID of the fork of SPOMM-MGAv3 that spawned the SPOMM run, it is used to see if the data from that run can be found. If you are generating this file yourself, and don't have the data, an arbitrary value can be used here. In the square brackets, the sample genome is given as a comma-separated list of key = value pairs. The keys are the names of SPOMM parameters and beta vector files in the order they appear in the GA file (see section 2.1). The values are those to insert in the genome. In parentheses in a comma-separated list at the end of the line is the fitness vector, as used by the GA for population selection (i.e. including any conflation). This is ignored if the original run data are not available.

5 Utilities

5.1 `setup-speciesdist2.pl`

This is a utility to help set up the files you need to run SPOMM-MGAv3. It is designed to assist in creating experiments for using SPOMM-MGAv3. For this, it is important to understand the difference between 'god' mode and 'mortal' mode. In 'god' mode, which will not be the normal case for running SPOMM-MGAv3, the 'true' habitats are known in advance, and this script can be used to create an occupancy distribution with the also known species α, μ, and x parameters for SPOMM-MGAv3 to fit.[6] In 'mortal' mode, only the biophysical characteristics and occupancy are known, and this script can be used to create SPOMM and biophysical characteristics files to supply as command line arguments to SPOMM-MGAv3, including (via a call to `enspline.R`) basis matrix, if the `searchSpline*BETA` sample rule is being used.

5.1.1 Synopsis ('mortal' mode)

[6] Note this sort of assumes that these are the species parameters that will be searched; hence the searchSPOMM section of the GA file will have lines for `SPOMM / xparam`, `species / ALPHA` and `species / MU`, as illustrated in section 2.1.

setup-speciesdist2.pl [-toroidal] [-splines <degrees of freedom>] [-biocols <

The experiment name simply provides a prefix for the files created by the script, and the biophysical variables file contains the biophysical variables. This is expected to be a text file, with columns for the x and y co-ordinates of each cell, and for the value of each biophysical characteristics variable in that cell. There is one line in the text file for each cell, and the file should start with a column heading line. White space separates entries in this file, hence column headings may not contain spaces. There are the following options:

`-toroidal` Stipulates a toroidal environment in the SPOMM file instead of the default bounded planar topology.

`-splines` Create biophysical characteristics files with basis matrix columns added with the given number of degrees of freedom.

`-biocols` Provide a comma-separated list of column headings for biophysical characteristics variables to use. (The default is to assume all columns not headed with the x or y co-ordinate column headings are biophysical characteristics variables.)

`-xcol` Specify the x co-ordinate column heading (default '`x`')

`-ycol` Specify the y co-ordinate column heading (default '`y`')

`-enspline` Provide a non-default location for the `enspline.R` script (only needed if the `-splines` option is used).

5.1.2 Output ('mortal' mode)

The script creates the following files:

expt_name`-biophys-`*var_name*`.csv` The biophysical characteristics files to pass to SPOMM-MGAv3. There will be one file per biophysical characteristics column, with *var_name* corresponding to the column heading for that variable in the biophysical variables file.

expt_name`-species.csv` A species file, with default values for species parameters. You should edit this file so that it matches your expectations for the variables you will not be asking the GA to search.

expt_name`.spom` A SPOMM file, for you to pass to SPOMM-MGAv3. As per the species file, you should edit this file so that it matches your expectations for the variables you will not be asking the GA to search.

5.1.3 Synopsis ('god' mode)

setup-speciesdist2.pl [-prob <steps>] [-sample <nspp>] [-seed <seed>] [-toroi

The additional command line arguments specify the location of the habitat file (which may be the same as the biophysical characteristics file if the `-habcol` and `-biocols` options are used), the species parameters α, μ, and x to use, and the number of steps to run the SPOMM before measuring occupancy. The command-line options in addition to those available in 'mortal' mode are as follows:

`-prob` Sample the probability of occupancy over the number of steps specified rather than just whether or not the species is present

`-sample` Repeat the sample the specified number of times.

`-seed` Stipulate a seed to run the SPOMM with

`-habcol` Give a name for the habitat column in the habitat file (by default, it is assumed to be the first column after whichever is the later of the x and y co-ordinate columns).

`-spom` Give a non-default location for spom-2.3.

The habitat file has the same format as the biophysical variables file in 'mortal' mode.

5.1.4 Output ('god' mode)

In addition to the outputs for 'mortal' mode, in 'god' mode, all the files needed to run the SPOMM are created, but the main output is the occupancy file:

`expt_name``-sppdist.txt` This file has a text format, with columns specifying the patch, and for each sample, the occupancy, either as 1 or 0, or if the `-prob` command line option is given, as a number representing the proportion of steps for which the patch was occupied. (Note that SPOMM-MVAv3 will only pay attention to the last sample column.)

A CSV format summary of the occupancy is also printed to standard output. Plotting this using R can be useful in determining the equilibrium status of the occupancy during the course of the SPOMM run. For this reason, recommended usage is to redirect the output of the script to *`expt_name`*`-sppdist.csv`.

5.2 `enspline.R`

This is a simple R script to create basis matrices for a series of biophysical characteristics files.

5.2.1 Synopsis

```
enspline.R <degrees of freedom> <biophysical characteristics files ...>
```

Here, degrees of freedom is the number of knots plus one, and biophysical characteristics files is a list of the biophysical characteristics files to convert. Each biophysical characteristics file is expected to be a two-column headed CSV file in the format specified in section 2.3.

5.2.2 Output

The script overwrites the biophysical characteristics files, adding columns containing the basis matrix (the result of running the R function `ns(data[,2], df = dof)`, where `dof` is the first command-line option and `data` is the table loaded in from the biophysical characteristics file).

References

[tB04] Cajo J. F. ter Braak. Genetic algorithms and markov chain monte carlo: Differential evolution markov chain makes bayesian computing easy. Technical Report 010404, Biometris, Wageningen University and Research Centre, 2004.

[tB06] Cajo J. F. ter Braak. A markov chain monte caro version of the genetic algorithm differential evolution: Easy bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249, 2006.

[tBV08] Cajo J. F. ter Braak and Jasper A. Vrugt. Differential evolution markov chain with snooker updater and fewer chains. *Statistics and Computing*, 18(4):435–446, 2008.