GMSE: an R package for generalised management strategy evaluation

Supporting Information 1

A. Bradley Duthie, Jeremy J. Cusack, Isabel L. Jones, Erlend B. Nilsen, Rocío Pozo, O. Sarobidy Rakotonarivo, Bram Van Moorter, and Nils Bunnefeld

2017-11-06

Extended introduction to the genetic algorithm applied in GMSE

A genetic algorithm is called in the predefined GMSE manager and user models to simulate human decision making. As of GMSE version 0.3.1.9, this includes one independent call to the genetic algorithm for each decision-making agent in every GMSE time step. Therefore, one run of the genetic algorithm occurs to simulate the manager's policy setting decisions in each time step (unless otherwise defined through non-default `manage_freq` values greater than 1), and one run occurs to simulate each individual user's action decisions in each time step (unless otherwise defined through non-default `group_think = TRUE`, in which case one user makes decions that all other users follow identically. Each run of the genetic algorithm mimics the evolution by natural selection of a populaiton of potential manager or user strategies over multiple generations, with the highest fitness strategy in the terminal generation being selected as the one that the manager or user decides to implement. For clarity, as in the main text, we use 'time step' to refer to a full GMSE cycle (in which multiple genetic algorithms may be run) and 'generation' to refer to a single, non-overlapping, generation of potential strategies that evolve within a genetic algorithm (see Figure 1 of the main text). Below, we explain the genetic algorithm in extended detail as it occurs in GMSE v0.3.1.9 (future versions of GMSE might expand upon this framework). We first explain the key data structure used, then provide an overview of the processes of crossover, mutation, cost constraint, fitness evaluation, tournament selection, and replacement. We then explain the fitness functions of managers and users in more detail.

Key data structures used

The focal data structure used for tracking manager and user decisions is a three dimensional array, which we will call `ACTION` (also returned as `user_array` by `gmse_apply`). Rows of `ACTION` correspond to recipients of actions (resources, landscapes, or potentially other agents), and columns correspond to either recipient properties or actions allocated to recipients. Each layer of `ACTION` corresponds to a unique agent, the first of which is the manager; additional layers correspond to each user. Below shows an `ACTION` array, which corresponds to a GMSE model with one manager and two users.

```
## , , Manager_Actions
##
##            Act Type_1 Type_2 Type_3     Util. U_land U_loc. Scare Cull
## Resource   -2      1      0      0 1000.0000      0      0     0    0
## Landscape  -1      1      0      0    0.0000      0      0     0    0
## Res_cost    1      1      0      0 -496.5986      0      0    10   10
## U1_cost     2      1      0      0    0.0000      0      0     0    0
## U2_cost     3      1      0      0    0.0000      0      0     0    0
##            Castrate Feed Help_off None
## Resource          0    0        0    0
## Landscape         0    0        0    0
## Res_cost         10   10       10  110
## U1_cost          0    0        0    0
## U2_cost          0    0        0    0
##
## , , User_1_Actions
##
##           Act Type_1 Type_2 Type_3 Util. U_land U_loc. Scare Cull Castrate
## Resource   -2      1      0      0    -1      0      0     0   68        0
## Landscape  -1      1      0      0     0      0      0     0    0        0
## Res_cost    1      1      0      0     0      0      0     0    0        0
## U1_cost     2      1      0      0     0      0      0     0    0        0
## U2_cost     3      1      0      0     0      0      0     0    0        0
##           Feed Help_off None
## Resource     0        0   17
## Landscape    0        0   15
## Res_cost     0        0    0
## U1_cost      0        0    0
## U2_cost      0        0    0
##
## , , User_2_Actions
##
##           Act Type_1 Type_2 Type_3 Util. U_land U_loc. Scare Cull Castrate
## Resource   -2      1      0      0    -1      0      0     0   75        0
## Landscape  -1      1      0      0     0      0      0     0    0        0
## Res_cost    1      1      0      0     0      0      0     0    0        0
## U1_cost     2      1      0      0     0      0      0     0    0        0
## U2_cost     3      1      0      0     0      0      0     0    0        0
##           Feed Help_off None
## Resource     0        0   12
## Landscape    0        0   13
## Res_cost     0        0    0
## U1_cost      0        0    0
## U2_cost      0        0    0
```

The above array holds all of the information on manager and user actions in the six right-most columns of each array layer. The first seven columns contain information about which resources are affected, and how they are affected. The first column `Act` identifies the type of action being performed; a value of -2 defines a direct action to a resource (e.g., culling of the resource), and a value of -1 defines direct action to a landscape (e.g., increasing yield). Positive values are currently only meaningful for `Manager_Actions`, where a value of 1 defines an action setting a uniform cost of users' direct actions on resources (i.e., costs where `Act = -2` for `User_1_Actions` and `User_2_Actions`). All other values for `Act` are meaningless in GMSE 0.3.1.7, but might be expanded upon in future versions to allow for modification of specific user costs enacted by managers (i.e., managers having different policies for different users) or other users (e.g., users increasing the costs of other users' actions due to conflict or cooperation). Similarly, columns 2-4 refer to resource or landscape types, but only `Type_1 = 1`, `Type_2 = 0`, and `Type_3 = 0` are allowed in GMSE v0.3.1.7 (i.e., only one type of resource is permitted), but future versions might allow for different resource types (e.g., `Type_1` might be used to designate species, and `Type_2` and `Type_3` could designate stage or sex). For the rest of this supporting information, we will therefore focus only on rows 1-3 of `ACTION`. Column 5 `Util.` of `ACTION` defines the utility associated with the resource (where `Act = -2`) or landscape (where `Act = -1`). For managers, the target resource abundance set with GMSE argument `manage_target` is found in row 1 (1000 in `ACTION` above); for users, the value in row 1 identifies whether resources are preferred to increase (if positive) or decrease (if negative). Values of column 5 in row 2 similarly identifies whether landscape cell output is preferred by users to increase or decrease (managers do not currently have preferences for landscape output). Of special note is row 3 for `Manager_Actions`, which defines the marginal utility of resources; that is, the adjustment to resource abundance that the manager will attempt to make based on the `manage_target` and the estimated abundance produced by the observation model (in the case of the above, resource abundance is estimated at ca 1065.76, so the manager will set policy in attempt to reduce the population size by ca 65.76 resources). Column 6 `U_land` defines whether or not the utility attached to the resource or landscape output depends on it being on a landscape cell that is owned by the acting user. Related, column 7 `U_loc.` defines whether or not actions can be performed only on a landscape cell that is owned by the acting user. Hence values of columns 6 and 7 are binary, and affected by the `land_ownership` argument in `gmse`. Finally, columns 8-13 correspond to specific actions, either direct (where `Act < 0`) or indirect by setting policy (for row 3 of `Manager_Actions` where `Res_cost = 1`). The last column 13 `None` corresponds with no actions. See GMSE documentation for details about the effects of each action.

General overview of key aspects of the genetic algorithm

Crossover

Mutation

Cost constraint

Fitness evaluation

Tournament selection

Replacement

Detailed explanation of manager and user fitness functions

Manager fitness function

User fitness function

Thanks for the clarification regarding the equation. I'll try to answer as best as I can – apologies if this has been unclear. At the broadest scale, the equation for user fitness would be on L367 in the strategy_fitness function (https://github.com/bradduthie/gmse/blob/master/src/game.c#L376). Here's what's going on: Users are predicting how their actions will change the quantities of things in the model (either resources or landscape output), and these changes are individually multiplied by the users' utilities for that thing. The change multiplied by utility for each thing is summed across all things to get a value for fitness. Note that positive change times positive utility, and negative change times negative utility, will increase fitness (i.e., increasing the thing users want more of and decreasing the things they want less of). Hence, an equation describing user fitness would be the below,

$$F_{user} = \sum_{i=1}^{N} \Delta A_i \times U_i$$

.

Where F_{user} is user fitness, N is the total number of things that might be of interest (at the moment $N = 2$ in GMSE, one resource and, potentially, one landscape value), ΔA_i is the change in the abundance of thing i, and U_i is the utility of thing i from the perspective of the user (apologies for the LaTeX code – attached a PNG of the conversion). I want to stress though that I would not consider this equation to be central to the GMSE framework – if someone else has a better approach for defining fitness, or defining any of the terms listed above, or wants to expand upon it to include new things, then that would be awesome! The above just works well as a heuristic tool to get users to act in such a way as to maximise their interests in harvesting or getting more crop yield (as is my intent), but it's not based on first principles and I don't claim it to be particularly special.

The values of ΔA_i are calculated for resources and the landscape in the functions res_to_counts and land_to_counts, respectively (and U_i is specified a priori in the model depending on other parameters – namely land_ownership). Again, a bit of heuristic is needed here because there cannot be any perfect way of exactly predicting how a users actions will increase or decrease resources – there are too many complex factors (e.g., behaviour of other stakeholders, demographic stochasticity, movement of resources on the landscape, and interactions between resources and the landscape). Even if we could include all of these things somehow, it would be a bit unrealistic in that real stakeholders would never have this much information. The predicted direct effect of actions on resources is shown in lines 268-272 (https://github.com/bradduthie/gmse/blob/master/src/game.c#L268), and the array 'jaco' (a sort of Jacobian matrix) accounts for interactions between landscape and resources on line 286. Something similar happens in the land_to_counts function. The manager's genetic algorithm works in a similar way (the above equation applies), but with the need to dynamically update utility values based on current resource abundance, and to account for the predicted actions of users in finding ΔA_i.