Urban Model: Development notes

Marta Vallejo
Heriot-Watt University
mv59@hw.ac.uk

April 1, 2015

1 General Description

1.1 Modules & Files

- Random: Random approach, non-optimised.
  - Baselines.txt
  - SatisfactionRan.txt
- GatherData: Used only to gather the statistical data.
  - Density.txt
  - NonUrbanPrices.txt
  - Urbanised.txt
  - Rings.txt
- ClosestHeuristic: The closes cell that we can buy (from Scenario 7)
  - Baselines.txt
  - SatisfactionClo.txt
- Genetic algorithm: GA_Satisfaction and GA_Distance.
  - GA_DATA.txt General info
  - Result_GA.txt with all the cells protected
- TestOptimisation: Test the GA approach.
  - SatisfactionGA.txt
  - Inconsistencies.txt
- MultiOptimisation: online GA Satisfaction
  - SatisfactionMO.txt
- CheckFiles:
  - Check Visually the distribution of cells protected in *Result_GA.txt* or *Protected.txt*
  - Check *Result_GA.txt* regarding the budget.
  - Visualise the population distribution in a given tick.

1.2 Variables

Seven kind of prices, each with:

1. Fitness:
   - Distance **only GA**
     - $Tick_0 - Min(Tick, sat) \neq 0$
     - $Min(Tick, sat) \neq 0$
   - Satisfaction
     (a) Ticks counted **only GA**
     - Accumulative. $\sum_{i=1}^{S} \sum_{j=1}^{C} \sum_{k=t_j}^{N} s(c_{i,j})_k$
     - Single.
       * Initial $\sum_{i=1}^{S} \sum_{j=1}^{C} s(c_i)_{t_0}$

    - End $\sum_{i=1}^{S} \sum_{j=1}^{C} s(c_i)_{t_N}$
  (b) Per agent
    - Unique: Maximum satisfaction per agent: $\forall a \in A, s(a) = \max s(a, l)$
    - Aggregated: Total satisfaction per agent: $\forall a \in A, s(a) = \sum s(a, l)$
      - Flat: only one point per park if it is at maximum 3 cells of distance.
      - Weighted: (1 to 3) point depending of the distance.

2. Selection creation process: **only GA**

   - Fix: all possible cells.
   - Stochastic: only a percentage of possible cells.

1.3 Files

Data gathered for 600 ticks of the clock.

1.3.1 Gather Data

For each time step, we store an entire lattice 50x50 cells (2500). The lattice is made up by a set of integer values. Each cell store a value in the system.

- **Non urban prices.**
  - The number of times that this row has been updated.
  - Mean of the price of the land when city grows.
- **Urbanised**
  - Number of times this row has been updated.
  - If the cell has been urbanised to the time t, then sum one in the lattice correspondent to this time step.

$$1 + SizeLattice$$

- **Density**
  - The number of times that this row has been updated.
  - If the cell is urbanised then it is calculated and stored the mean of the previous value and the new value. Change: only sum and it can be divided later.
- **Rings** Non-urban averaged priced in the different rings of the lattice.

1.3.2 TestSolution, Random, Closest Heuristic & Multioptimisation

TYPE_FITNESS: S(Satisfaction), D(Distance).
TYPE_SELECTION: F(Fix), S(Stochastic).
TYPE_SATISFACTION_AGENTS: Flat: 1 / Weighted: 2
TYPE_FITNESS_TICKS: Accumulative (10), Single - Initial: (21) / End: (22) *TYPE_OPTIMISATION*: 1, TODO MixApproach *SAVE_DENSITY*: 1 Save population distribution to calculate real fitness.

- **Satisfaction** Different runs are appended into the same file if the run is not finished properly. Each line corresponds to one time step.

  1. Total population.

2. Total satisfaction (1): All the green spaces count.
3. Total satisfaction (2): Only the closest green space counts.
4. Total urban cells.
5. Total protected cells.
6. Min Green price.
7. Max Green price.
8. Average Green price
9. Min Urban price.
10. Max Urban price.
11. Average Urban price
12. Min populated
13. Max populated
14. Migration
15. Closeness to the CBD
16. Protected cells

SatisfactionMO (fields added):

1. Time
2. Protected Cells
3. GA Population size
4. Generations until convergence
5. Number of mutations
6. Worst fitness
7. Best fitness

In SatisfactionGA (name file):

1. GA ID (4 digits).
2. Satisfaction ID (4 digits).
3. Type of fitness: (*TYPE_FITNESS*)
   (a) Satisfaction (S) - Both aspects
      – Ticks (*TYPE_FITNESS_TICKS*)
         ∗ Accumulative (10)
         ∗ Single - Initial: (21) / End: (22)
      – Agents - Flat: 1 / Weighted: 2 (*TYPE_SATISFACTION_AGENTS*)
   (b) Distance (D)
4. Type of selection. (*TYPE_SELECTION*)
   (a) Fix (F)
   (b) Stochastic (S) - <percentage> (*PERCENTAGE_STOCHASTIC*)
5. Feasibility
   (a) Feasible (F)
   (b) Infeasible (I)

Example: 1234 5678 S101 F I - 1234 5678 D S50 F

In SatisfactionMO (name file):

1. GA ID (4 digits).

2. Satisfaction ID (4 digits).
3. Type of fitness: S (*TYPE_FITNESS*)
   - Ticks (*TYPE_FITNESS_TICKS*)
     - Accumulative (10)
     - Single - Initial: (21) / End: (22)
   - Agents - Flat: 1 / Weighted: 2 (*TYPE_SATISFACTION_AGENTS*)
4. Type of selection. (*TYPE_SELECTION*)
   (a) Fix (F)
   (b) Stochastic (S) - <percentage> (*PERCENTAGE_STOCHASTIC*)

Example: 1234 5678 S101 F - 1234 5678 S101 S50

- **Budget** Budget used to buy and protect land. One per time step. Field: Amount of budget given to the municipality in each time step. It is generated randomly.
  - BUDGET_R: Randomly generated
  - BUDGET_P: In function of the population
- **Scenario** Initial biovalues of the cells included into the lattice. Type: double.

1.3.3 GA Phase

- **ResultGA** Cells selected to be protected according to the GA optimisation procedure.
  File content:

  1. Tick of the clock only when a cell is protected: it can be repeated if two cells are protected in the same slot.
  2. Pair/s of coordinates of each cell.
  3. Price of the cell.

  Name Format:

  - Times that statistics are gathered.
  - Type of fitness: (*GA_Satisfaction* or *GA_Distance projects*)
    1. Satisfaction (S) - Both aspects
       - Ticks (*TYPE_FITNESS_TICKS*)
         - Accumulative (10)
         - Single - Initial: (21) / End: (22)
       - Agents - Flat: 1 / Weighted: 2 (*TYPE_FITNESS_AGENTS*)
    2. Distance (D)
  - Type of selection.
    1. Fix (F)
    2. Stochastic (S) - <percentage>
  - Feasibility
    1. Feasible (F)
    2. Infeasible (I)
  - ID (Four digits).

  Example: 20 S101 F F 1234, 20 S212 S50 I 1234, 20 D F I 1234, 20 D S50 F 1234

- **GA_DATA** Located in folder *Workspace* to be updated from both GAs.

1. Type optimisation: OFF_SAT, ON_SAT
2. Lattice size
3. GA Population size.
4. Ticks of the simulation.
5. Times that statistics are gathered.
6. Generations until convergence.
7. Number of mutations.
8. Worst fitness.
9. Best fitness.
10. Mean fitness
11. Standard Deviation SD
12. ID (Same than Result_GA).
13. Type of fitness:
    (a) Satisfaction (S) - Both aspects
        – Ticks
            * Accumulative (1) - **0**
            * Single (2) - Initial: 1 / End: 2
        – Agents - Flat: 1 / Weighted: 2
    (b) Distance (D) - **0 - 0 - 0**
14. Type of selection.
    (a) Fix (F) - **0**
    (b) Stochastic (S) - <percentage>
15. I/F Infeasible solutions - Feasible solutions
16. Scenario (Type of prices).
17. CBDs.
18. Type of Budget
19. Computational Time

Example:15 600 20 xxxx xxxx xxx xxx 1234 S 1 0 1 F 0 9g, ...1234 D 1 0 1 S 50 9F

Mean and SD fitness should be calculated in MATLAB

- GA Parameters

1. Number of individuals in the population.
2. Number of simulations.
3. Size of the lattice.
4. Scenario.
5. CBDs info.
6. Feasible - infeasible solutions.

1.3.4 TestOptimisation

- **Inconsistency** One file for all runs.

1. GA ID (4 digits).
2. Satisfaction ID (4 digits).
3. Number of inconsistencies: times that we try to protect a cell that is already urbanised.
4. Number of failures because of a lack of budget.
5. Times statistics are collected.
6. Type of fitness:
   (a) Satisfaction (S) - Both aspects
      – Ticks
         * Accumulative (1) - **0**
         * Single (2) - Initial: 1 / End: 2
      – Agents - Flat: 1 / Weighted: 2
   (b) Distance (D) - **0 - 0 - 0**
7. Type of selection.
   (a) Fix (F) - **0**
   (b) Stochastic (S) - <percentage>
8. Feasibility
   (a) Feasible (F)
   (b) Infeasible (I)
9. Scenario.
10. CBDs.
11. Size lattice
12. Total Time
13. Type Price
14. Type Budget

Example: 1234 5678 1 15 20 S 1 0 1 F 0 9g, xxx D 0 0 0 S 50 9f

1.3.5 MOOptimisation

- **Inconsistency** One file for all runs.

1. GA ID (4 digits).
2. Satisfaction ID (4 digits).
3. Type of fitness:
   (a) Satisfaction (S) - Both aspects
      – Ticks
         * Accumulative (1) - **0**
         * Single (2) - Initial: 1 / End: 2
      – Agents - Flat: 1 / Weighted: 2
   (b) Distance (D) - **0 - 0 - 0**
4. Type of selection.
   (a) Fix (F) - **0**
   (b) Stochastic (S) - <percentage>

5. Scenario.
6. CBDs.
7. Size lattice
8. Time

Example: 1234 5678 1 15 20 S 1 0 1 F 0 9g, xxx D 0 0 0 S 50 9f

- **Baselines**

1. CLO/RAN
2. id
3. Scenario name
4. CBDs
5. Size lattice
6. Budget
7. Total time

1.4 Scenarios

1.4.1 Backup Allocation

At University: *mv59/private/versions*
Home: *Documents/PhD/CollectedData/Code*

2 How to run experiments: Steps

All the programs should share the same *scenario.txt* file and *budget.txt*. Create a folder with the name of the new scenario if it is necessary and place the files into the folder: CollectedData/Calculos.

- In the GatherData module.
  - Check current size of the lattice and change it if it is appropriate.
  - If we want to start from scratch, initialise Urbanised, NonUrbanPrices & Density files to zero.
    - Location of the files: collectedData/Scenarios/CaseBase/FilesToZero.
    - Take the ones with the correct size of the lattice and with the proper number of ticks of the clock.

    Otherwise copy them form the case we want to continue.
  - In *lattice.java*
    - update *NUM_CBDS* and *CBDS* according to the desired scenario.
    - update scenario to assign the proper non-urban prices in *cell.java*.
  - Run *nLattice::TOTAL_TICKS* times GatherData to collect statistical data for the GA in the Urbanised, NonUrbanPrices & Density files.
    - Create a folder with the time that the data is gathered.
- In the GA Satisfaction/Distance project (normal Java project with parameters).
  - Check input parameters: population size, number of generations, size lattice and scenario name.
  - Update Density, NonUrbanPrices & Urbanised files with the new files calculated before.
  - Run the program.

  - Copy *Result_GA.txt* to CollectedData/Scenario/NomScenario/Num_Runs when the program finishes

- The file *Workspace/GA_Data.txt* is automatically updated.
- In TestOptimisation project:
  - Parameters: size of lattice: in <*projectname*>.*rs context.xml*::"World Size", "width", "height" (source view).
  - In *lattice.java* update:
    - Number of runs *(TOTAL_TICKS)*
    - *NUM_CBDS* and *CBDS* according to the desired scenario.
    - scenario to assign the proper non-urban prices in *cell.java*.
    - *SCENARIO_NAME* with the name given to the current scenario.
  - Copy the *Result_GA.txt* file from GA that we want to check.
  - Run the program
  - Copy *satisfaction.txt* into the folder CollectedData/Scenarios/NomScenario/Num_Run.
  - The file *inconsistecy.txt* is automatically updated.
- When the collection of data is finished for the current scenario, copy *GA_Data.txt* and *inconsistecy.txt* to CollectedData/Scenario/NomScenario
- In Random project & Closest project
  - Check parameters: size of lattice
  - In *lattice.java* update:
    - number of runs *(Lattice::TOTAL_TICKS)*
    - *NUM_CBDS* and *CBDS* according to the desired scenario.
    - scenario to assign the proper non-urban prices in *cell.java*.
  - Run Random / Closest
  - Place & *satisfactionRAN.txt* into CollectedData/Scenarios/NomScenario/RAN or *satisfactionCLO.txt* into CollectedData/Scenarios/NomScenario/CLO as corresponding
  - *protected.txt* is not currently used.

If there are enough changes to create a new scenario, copy the *.java* files into folders, zip them and copy to the folder *CollectedData/Code*

2.1 List of Functions in Matlab

- **budgetGenerator** Generate a file budget dependant on the population alpha = 0.5: measure the importance of the population when the budget is generated. Density.txt: File with population evolution
- **simpleSatisfaction** Show simple satisfaction comparative for GA, MO, CLO and RAN
- **simpleSatisfactionBar** Show simple satisfaction comparative for GA, CLO and RAN
- **satisfactionTable50** Show simple satisfaction comparative for GA and RAN
- **latexTable** Create the code in latex for a table with the satisfaction achieved by the three approaches
- **bestSatisfactionTest** Create a plot with the best Test values achieved. **It doesn't work**
- **failuresPlot** Create a stacked bar chart using the bar function. **It doesn't work. Inconsistencies has change**
- **populationPlot** Create a plot with the population behaviour

- **cellsUrbanisedPlot** Create a plot with the number of cells urbanised
- **protectedCellsPlot** Create a plot with the number of cells protected
- **migrationPlot** Create a plot with the behaviour of the migration
- **lowerGreenPricesPlot** Create a plot with the collected green prices with the lowest value
- **higherGreenPricesPlot** Create a plot with the collected green prices with the highest value
- **greenSpacesPlot3D** Test the 3-D shaded surface plot for collected green prices
- **lowestUrbanPrices** Create a plot with the urban cells with the lowest value
- **higherUrbanPricesPlot** Create a plot with the urban cells with the highest value
- **avgGreenPricesPlot** Create a plot with the average of green prices
- **avgUrbanPricesPlot** Create a plot with the urban cells with the average value
- **GADataPlot** Create a plot with the GA data. **It doesn't work. GA_Data file has change**
- **greenPricesPerRing** Plot average green price prices grouped by rings. Place Ring.txt in General folder before run the function
- **Ring_GreenPricesPlot** Double Plot: Plot average green price prices grouped by rings. *In which tick of the clock?*. Average green prices in GA
- **exponentialFunction** Return the exponential function of the non urban prices data
- **satisfaction_protected_closeness**ICCS plots: 3subplot: satisfaction, number of cells protected and closeness
- **satisfactionArea** Area of the satisfaction of three heuristics. **TODO:** Collect the data automatically from the three scenarios
- **OnOfCellsProtectedPlot** Plot with the number of cells protected for Online/Offline/Mix
- **OnOfSatisfaction** Create a line plot with the satisfaction for Online/Offline/Mix
- **OnOfCloseness** Create a plot with the closeness to CBD for Online/Offline/Mix
- **OnOfTiming** Computational time for Online/Offline/Mix. **TODO.** Not really implemented

3 Differences between Scenarios

3.1 First attempts

- Scenario 1: Changes in the way cells are selected in GA. Fix a problem with the size of the CA in the check scenario
- Scenario 2. Add new statistic material
- Scenario 3. Fixed the growth of the city
- Scenario 4. Fixed a problem in gathering the position of people.
- Scenario 5, 6. Collected data without protection of cells. Fixed a problem in GA positions.
- Scenario 7 they depend on the distance.
  - **Fixed:** Calculating fitness values in GA. Now the protection of cells is homogeneously done.
  - **Add:** CHANGE_RATE constant = 0.2.
  - **Change:** Non-urban prices depend on the distance to CBD.
  - **Add:** Var lastUrbanised, getLastUrbanised() and setLastUrbanised()

3.2 Scenario created for the ICAART journal

3.2.1 Scenario 8

- **Add:** New source of uncertainty: non-urban prices are not constant.
- **Remove:** Concept of tolerance and threshold (Genetic algorithm)
- **Add:** New form to accept a cell to be protected, urbanisation factor (Genetic algorithm)
- **Change:** Calculation of fitness (linked with urbanisation factor).
- **Fixed:** Calculation of position (Lattice, GA)
- **Fixed:** Error found in TestOptimisation::Cell::ReduceDemand

3.2.2 Scenario 9

- **Fixed:** Forest and agricultural price was swapped
- **Fixed:** *GatherData::CHANGE_RATE* was 0.8 and not 0.2.
- **Fixed:** Urban prices which gives higher values in green prices.
- **Add:** Gather statistics about cells (min, max, avg).
- **Add:** Gather statistics about the GA algorithm (max and avg fitness, mutations).
- **Fixed:** Times the mutation procedure is tried in the GA.
- **Fixed:** Statistics files were written in gather data differently than were read in GA.
- **Fixed:** Satisfaction was calculated differently in GA than in the rest of the modules.
- **Modified:** Price recent development is calculated taking all the cells in the outer annulus instead of only the last cell urbanised.

3.3 Scenario created for ICCS2015

3.3.1 Scenario 10

- **Add:** More than one CBD in the simulation.
- **Add:** Capacity to choose among different fitness distances.
- **Add:** New stochastic way of generating the selections in GA.
- **Add:** Statistics gathered added automatically to *inconsistecy.txt* & *DATA_GA.txt*.
- **Add:** Information to the name of the files during all the process.
- **Add:** GA feasible & infeasible solutions.
- **Mod:** How protected cells are managed in testing
- **Fix:** Gather data shift one position the statistical data gathered
- **Add:** Checkfiles: Visualise population
- **Fix:** prices constant when they shouldn't in GA_SAT
- **Fix:** Duplicated cells to be protected
- **Fix:** One of the three cities creates cells further from CBD

3.3.2 Scenario 11

- **Modified:** Size of the lattice equal to 100.

3.3.3 Scenario 12

- **Modified:** Budget linked to population growth with the use of parameter alpha in Matlab.
  In GA_SAT: change TYPE_BUDGET = 1. Rest Lattice: TYPE_BUDGET = 1.

3.4 Scenario created for UAI2015

3.4.1 Scenario 13

- **Add:** Module MultiOptimisation where GA optimisation is done online.
- **Add:** Variable Individualxxxx: times a suitable individual is searched by the GA.
- **Add:** Time total and partial gathered (Multi).
- **Fix:** Cells were not removed from the list of NON_URBAN_CELLS when they were protected. Allow a cell be protected twice.
- **Add:** Baseline.txt file to collect general information of CLO and RAN like time, CBDs...
- **Fix:** Problem found when a cell was selected for urbanised in the same turn that it was protected.
- **Add:** Initial conditions are checked in lattice.

4 Future extensions

4.1 Benchmarking

Perform a benchmarking comparison analysis or benchmarking between a GA approach and other kind of techniques naturally more adequate to solver sequential-decision making problems like:

- Reinforcement Learning (RL). RL was formalised by Barto [1]and it consists of a machine learning technique capable of selecting the optimal policy and representing explicitly the uncertainty.
- Simulated Annealing
- Markov Chains
- Spatial logistic regression

Goal: check if our approach is the best for this particular problem. Comparison of heuristics that depends on:

- Problem formulation.
- Parameters specifically used for the method used: determine how much time is required to complete the search (computational cost)
- Time given to search throughout the search space and performance [4].

4.2 Improvements of the model.

- Satisfaction increases with the quality of the green area that can be measured by two factors: higher ecological value (preference for forest against agricultural land) and the extension of the protected area. Perform a pre-clustering of the areas.Measure how crowded the green areas are and penalise the satisfaction in case the area is overcrowded.
- Associate the green satisfaction factor and the fitness function with the individual willingness of living close to a green space.
- Include other metrics.
- Adequate the salary to a normal distribution.
- Study differences in prices of buying larger extensions of land (price & satisfaction behaviour).
- Fix the problem salaries too low at the end of the simulation. (Study which part of the salary is used to pay housing expenses). Divide agents into its profile [2].
- Implement change of residence if satisfaction is not enough. Redevelopment deactivate.
- Measure the influence of the initial scenario in the results.
- Most individuals do not have a complete knowledge of the possible set of available places. Include a stochastic factor: sometimes not the best areas are developed.
- Search better behaviour for migration.
- A non-homogeneous distribution of resources.

5 Notes

Directions in all modules:

$$dir = (x * size) + y$$

In CLO searchGreenSpaces is triggered by municipality and not by updateAggregate.

References

[1] Andrew G. Barto, Richard S. Sutton, and Peter S. Brouwer. Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 40:201–211, 1981. 10.1007/BF00453370.

[2] W. Loibl and T. Toetzer. Modeling growth and densification processes in suburban regionssimulation of landscape transition with spatial agents. *Environmental Modelling & Software*, 18(6):553 – 563, 2003.

[3] Dave Murray-Rust, Verena Rieser, Derek T. Robinson, Vesna Milii, and Mark Rounsevell. Agent-based modelling of land use dynamics and residential quality of life for future scenarios. *Environmental Modelling & Software*, 46(0):75 – 89, 2013.

[4] Timo Pukkala and Mikko Kurttila. Examining the performance of six heuristic optimisation techniques in different forest planning problems. *Silva Fennica*, 39(1):6780, August 2005.