Nature Inspired Optimisation for Delivery Problems
Chapter 6: Illuminating Problems

Neil Urquhart

May 27, 2022

These slides are designed to accompany the book ”Nature Inspired Optimisation for Delivery Problems : From Theory to the Real World”.

`https://link.springer.com/book/10.1007/978-3-030-98108-2`

1. Illumination Algorithms

2. Using MAP-Elites to Plan Deliveries

3. Summary

Illumination Algorithms

Introduction

- It may be argued that the most powerful tool in our quest to solve delivery problems is not computer hardware and software, but the knowledge and experience of a human expert.
- Techniques such as Evolutionary Algorithms [?] allow computers to become effective at the task of evaluating many solutions in a very short time span. If we require a solution that matches a set of well-defined criterion then such approaches will serve us well.
- In real-world scenarios the user may need to address multiple constraints, stemming from a range of organisational and political objectives and aspirations.

Introduction

- In the context of the on-demand economy, decisions may also have to taken rapidly as the time scale from a customer placing an order online to expecting delivery grows ever shorter.
- The time available for running (and re-running) algorithms grows shorter, this becomes acute if a last minute change of plan is called for due to issues such as staff shortages, weather or vehicle availability.
- For example the user may be faced with a commitment to make use of low-carbon delivery modes, but might also be under pressure to reduce operating costs. At that point the choice of final decision is perhaps best left to an experienced human expert.

Illumination Algorithms - MAPElites

- The construction of a non-dominated front gives the user a choice of solutions
- This has the advantage of incorporating the users' expertise and allowing for a level of decision making above the planning undertaken by the algorithm.
- This principle may be extended through the use of an *illumination algorithm*.
- An Illumination Algorithm seeks to find a set of high quality solutions that represent the entire solution space, giving the user a large number of solutions to choose from.

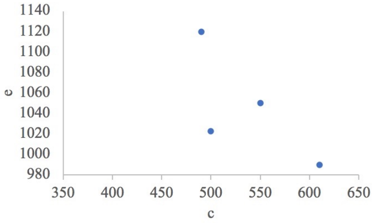
Illumination Algorithms - MAPElites

The Map Archive of Phenotypic Elites (MAP-Elites) was developed by Mouret et. al.

- MAPElites finds a structured set of high quality solutions from across the entire search space.
- The user can then use the set of solutions to make an informed choice of final solution
- The key requirements are :
    - that the solutions are held in a structure which allows a user to "browse" through them
    - that the solutions represent high quality (or in real-world terms, useful) solutions.
    - the set encompasses a diverse variety of solutions that illuminates the possibilities open to the user within the search space

A MapElites example

Consider a logistics problem that has 2 solution characteristics, financial cost c and environmental impact (CO_2) e. Suppose we evolved the following solutions to our problem:



Assuming that c and e are integers, We could calculate the number of possible solutions as $(650 - 350) * (1140 - 950) = 57,000$. We could attempt to create an archive of 57,000 solutions, but this has a number of drawbacks ...

A MapElites example. ... cont

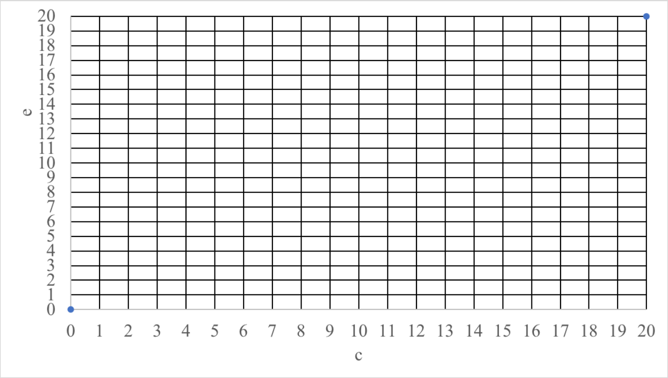
- The resources required to maintain a data structure of 57,000 solutions may impose a severe run-time overhead on our software
- Many of these 57,000 solutions may not be valid solutions
- Many of the solutions may be very similar to others
- Many of the solutions may be of a poor quality

The MAPElites Archive

- MAP-Elites represents the solution space with a smaller archive set of high quality (or *elite*) solutions.
- The MAP-Elites archive contains a series of "cells", every solution in the solution space maps to a cell. To achieve this, Map Elites scales each solution characteristic into a smaller range. We we might decided to scale our two objectives into the range 1-20.

MapElites Archive Example

An empty archive might look like this:



MapElites Archive Example ... cont

For any characteristic the raw value r can be translated into the scaled value s as follows:

$\delta = (max + 1) - min$

$cap = \frac{\delta}{b}$

$s = int(\frac{r - min}{cap} + 1)$

Where

- max is the maximum value of the characteristic
- min is the minimum value of the characteristic
- δ is the range of the characteristic
- cap is the size of each bin (the number of raw solutions encompassed by each bin)
- b is the number of bins

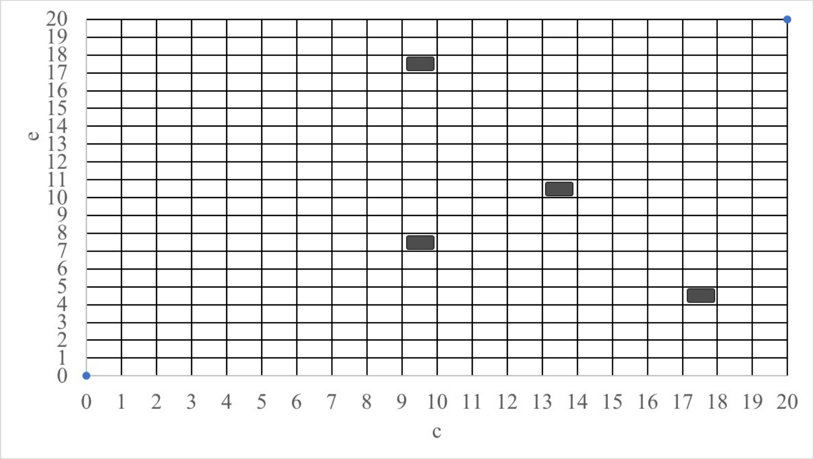
MapElites Archive Example ... cont

Using this formula our four solutions shown above would map to bins as follows:

Raw		Cell	
c	e	c	e
500	1023	10	8
550	1050	14	11
610	990	18	5
490	1120	10	18

MapElites Archive Example ... cont

Which would occupy the archive as follows:



MAPElites Archives in Detail

- Our example archive of 400 cells which is designed to cover a search space of $57,000$ possible solutions
- There are $\sim$142 actual solutions that map to each cell, so there is a significant chance that a solution will be generated that maps to a cell that is already occupied.
- We use a *fitness* value f to determine which solution should occupy a cell.

MAPElites Fitness

- Our example problem has the solution characteristics (c, e), in addition to this we now need a fitness value f.
- In this case we could use distance travelled as our fitness
- The overall distance travelled will have a relationship with c and e less distance should equate to less environmental impact and less cost.

The MAPElites Algorithm

```
Procedure MAP-Elites(init,totEvals,xOverPressure)
buckets = 20 dimensions = 2 evals = 0
map.initialise(buckets,dimensions)
while $evals < init$ do
    add(map,new Individual()
end
while $evals < totEvals$ do
    if $random() < xoverPressure$ then
        c = new Individual(map.random(),map.random())
    end
    else
        c = new Individual(map.random())
    end
    c.mutate()
    add(map,c)
end
return map
```

The MAPElites Algorithm

```
Procedureadd(map,n)
key = getKey(n)
if map.get(key) == null then
    map.put(key,n)
end
else
    old = map.get(key)
    if new.fitness()<old.fitness() then
        map.put(key,n)
    end
end
EndProcedure
```

Using MAP-Elites to Plan Deliveries

Case Study

- We will examine the problem from the previous chapter from the perspective of the planner who must find a solution that best matches the current business objectives and constraints,
- We will also add cargo bikes into the scenario.
- An updated model is required that identifies the costs associated with making deliveries.

Solution Characteristics

The solution characteristics that may be considered by the planner

	Solution Characteristic
1	The total daily fixed cost
2	The staff total staff cost
3	The total vehicle running cost
4	The cost per delivery (per crate)
5	The emissions produced
6	The % of deliveries made by bike
7	The % of the distance made by bike
8	The number of bicycles required
9	The number of vans required

Using MAP-Elites we can optimise taking into account all of the above and then allowing the manager to select a solution based on their requirements.

Implementation

Building on the EA used in the previous chapter we need a revised representation to take into account the preferred delivery mode (cycle or van).

An example chromosome might look like this :

5,0,V	2,0,B	4,0,B	8,0,B	1,0,B	7,1,B	3,0,B	6,0,B

Where each gene contains 3 items,

- **Customer id** 1-*n*
- **New Route** 0—1
- **Preferred Mode** V—B

Note that As we have two modes, we must have two sets of travelling times.

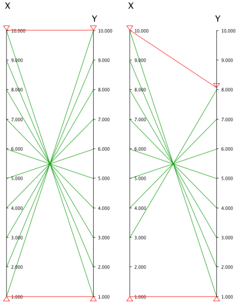
The Revised Decoder

- A revised decoder calculates the fitness and appropriate values for each of the solution characteristics, allowing solutions to be mapped to the appropriate cell within the archive.
- When a new route is commenced, the mode (Van or Cycle) is determined based on the preferred mode of the first customer

Setting The Ranges of the Solution Characteristics

- One aspect of MAPElites implementation that requires careful consideration is the setting of the maximum and minimum values required to normalise each solution characteristic.
- If characteristics x and y are opposed, setting the maximum value for x too low may make it impossible to find solutions with low values of y.

The effect of reducing the upper bound of Y (right).

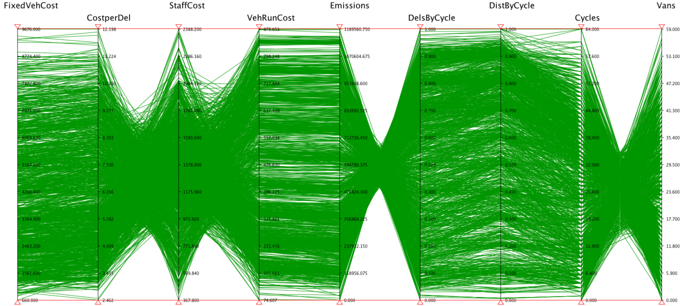


Supporting Solution Choice

- A convenient means of visualising the contents of the archive is the use of a *parallel coordinates* plot
- Each solution characteristic becomes a vertical axis
- Each solution is plotted using a poly-line that intersects the axis at the appropriate points

Parallel Coordinates

A MAPElites archive based around 9 solution characteristics:

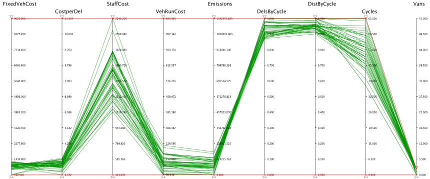


Parallel Coordinates

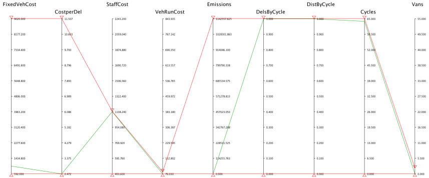
- When an archive contains many solutions it can be difficult to distinguish them on the PC plot.
- Using an interactive PC plot can allow the user to those solutions that pass through specific areas of the axis

Parallel Coordinates

Highlight solutions with low numbers of vans



Highlight solutions low numbers of vans AND low staff cost AND low vehicle running cost



Summary

Conclusions

- MAPElites allows us to change how we think about optimisation, instead of searching for a single or small number of solutions MAPElites attempts to fill an archive
- MAPElites treats all objectives equally when filling the archive
- An effective method of supporting the user when selecting the final solution from the archive is necessary (parallel coordinates).