Assessment Task 2

Stochastic Optimization using Population Based Incremental Learning

Professor Robin Braun

School of Computing and Communications
University of Technology Sydney

Semester 1 of Data Engineering

Outline

Motivation

- A very important element of Data Engineering is the ability to optimize things
- Very often this is done through Machine Learning
- The motivation for this task is to introduce Machine Learning / Stochastic Optimization using;
  - A simple to learn algorithm, on
  - A simple well know “benchmark” problem, using
  - A programming language that is already familiar (Matlab)
- Having said that, the Algorithm turns out to be a very powerful and useful one - good to have in your toolbox.

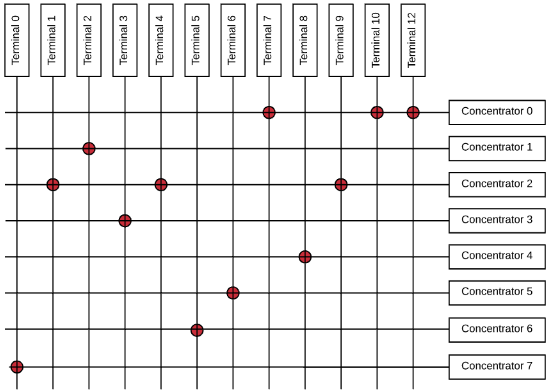
The benchmark problem

1. The problem chosen is the Terminal Assignment Problem
2. This is a classic “benchmark” used in many network planning courses
3. It has application in many areas, including such areas as Supermarket / Supply Depot allocation

The terminal assignment problem I

- This is the typical situation where we have a number of terminals connected to concentrators
    - or, for example Supermarkets supplied by Warehouses
- There is a “cost” associated with each of the potential connections
- In addition, there are always constraints that must be met. In this case
    - Each terminal must be connected to 1, and only 1 concentrator, and
    - There is a maximum number of terminals that each concentrator can service

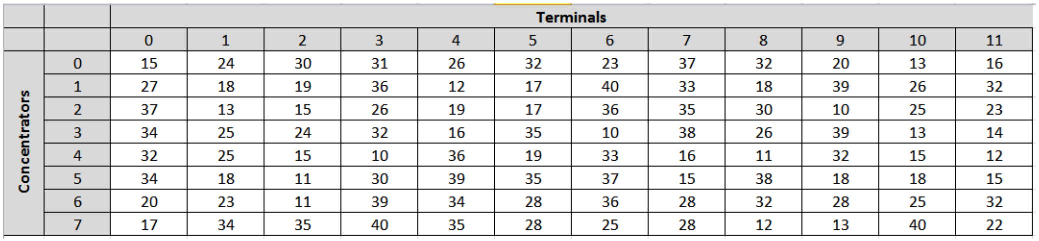
Terminal / Concentrator layout



- The layout show a number of connections
- There should be 12 as there are 12 Terminals
- In this case, Terminals 7, 10 and 11 are connected to Concentrator 0, which is quite legal.
- The total Cost of this configuration is the some of all the 12 values associated with the red dots. (See the Cost Table next.)

The Cost Table

To be explicit, if Terminal 5 is connected to Concentrator 3 the cost will be 35



		Terminals											
		0	1	2	3	4	5	6	7	8	9	10	11
Concentrators	0	15	24	30	31	26	32	23	37	32	20	13	16
	1	27	18	19	36	12	17	40	33	18	39	26	32
	2	37	13	15	26	19	17	36	35	30	10	25	23
	3	34	25	24	32	16	35	10	38	26	39	13	14
	4	32	25	15	10	36	19	33	16	11	32	15	12
	5	34	18	11	30	39	35	37	15	38	18	18	15
	6	20	23	11	39	34	28	36	28	32	28	25	32
	7	17	34	35	40	35	28	25	28	12	13	40	22

Mathematical Formulation

Connectivity Variable

To begin the mathematical description of the problem, we will define a variable x_{ij} that describes the connectivity

$$x_{ij} = \begin{cases} 1 & \text{terminal i connected to concentrator j} \\ 0 & \text{not connected} \end{cases}$$

Mathematical Formulation (cont..)

Cost variable

We can define another variable that defines the "cost" of such connections. The cost can be many things, or compounded factors. It may be the actual cost in $s, or the difficulty of making the connections, degradation in speed etc. etc.

$$c_{ij} = \text{Cost of connecting terminal i to concentrator j}$$

Mathematical Formulation (cont..)

Objective Function

- These tow variables get combined to for a formula which calculates the Cost of a particular connection matrix
- We call this the Objective Function.

$$Z = \sum_{i=1}^{N} \sum_{j \varepsilon J} c_{ij} x_{ij}$$

- Assume we wish to minimize the cost. So we are looking for $min(Z)$

Mathematical Formulation (cont..)

Constraints

- As in all such optimization problems, there are constraints.
- For example, we might wish to assert that each terminal can connect to only one concentrator. We can do that as follows:

$$\sum_{j \varepsilon J} x_{ij} = 1 \text{ for } i = 1, 2, \cdots, N$$

- We may also wish to limit the number of terminals that each concentrator can handle. We can do this as follows:

$$\sum_{i=1}^{N} x_{ij} \leq \text{q for } j \varepsilon J$$

Create a program that calculates *minZ*

- You need to use Matlab to write a program that performs the Genetic Algorithm.
- The Genetic Algorithm you need to use is called Population Based Incremental Learning (PBIL).
- Once your program is operational, you need to be able to change the program variables to find the optimum parameters needed to find a solution that is close enough to optimum.
- You need to reflect on the results, drawing conclusions as to your success.

Simple explanation of the PBIL Algorithm I

PBIL is a simple stochastic/genetic algorithm for optimizing a configuration. It requires the following:

1. A description of the solution space by means of a binary string
2. A probability vector that is the same length as the binary string, where each location indicates the likelihood of that binary digit in the solution string being a 1.
3. A means of calculating a "Cost" from that binary string using a Cost Table
4. The algorithm proceeds as follows:

Simple explanation of the PBIL Algorithm - Step 1

Step 1: Creating an epoch of sample solutions

1. Create a number of sample solutions based on the probability vector.
2. This is called an epoch and could have any number of solutions. Perhaps 100.
3. Each sample solution is another of the binary strings
    1. Say the Nth element of the Probability Vector is 0.5, then the Nth element of the binary string would be equally likely to be a "1" or "0"
    2. Say the Nth element of the Probability Vector is 0.667, then the Nth element of the binary string would be a "1" on 66% of the trials
    3. Say the Nth element of the Probability Vector is 1, then the Nth element of the binary string would always be a "1"
    4. Say the Nth element of the Probability Vector is 0.0, then the Nth element of the binary string would never be a "1"
4. And so on for all the elements of the binary string
5. Check that all the solutions found do not abrogate the constraints

Simple explanation of the PBIL Algorithm - Step 2

Step 2: Working out the costs

1. Now that you have an epoch of sample solutions, which are just binary strings, you have to calculate the Cost of each solution.
2. Use you cost table, with each sample solution to calculate the Cost of that solution

Simple explanation of the PBIL Algorithm - Step 3

Step 3: Ranking the epoch

1. Based on the costs calculated above, find the minimum one
2. You could do this by ranking them.
3. Now take the minimum one and use it in the next step

Simple explanation of the PBIL Algorithm - Step 4

Step 4: Updating the Probability Vector

1. You now have a binary string that represents the best solution of your epoch of possible solutions
2. Use it to modify the Probability Vector so that the next epoch you create will be more similar to this best solution found in Step 3.
3. So, if element N was a "1" then the corresponding Nth element of the Probability Vector needs to be adjusted by a small amount so that in the next epoch, the Nth elements of the sample solutions are more likely to be a "1".
4. And so on for all the elements

Simple explanation of the PBIL Algorithm - Step 5

Step 5: Go back and do it again

1. Return to Step 1: and create another epoch of solutions
2. Proceed through Steps 2, 3 and 4 as before
3. Inspect your Probability Vector.
4. It should start to look more like a string of "1s" and "0s" each time you go around the process.
5. When the Probability Vector has finally settled as a string of "1s" and "0s", you know you have a solution.

Detailed working of the algorithm - Start

Tabular representation

Probability Vector	P_0	P_1	P_2	$\cdots$	P_N	
Trial 0	$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$\cdots$	$B_{0,N}$	Cost of Trial 0
Trial 1	$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$\cdots$	$B_{1,N}$	Cost of Trial 1
Trial 2	$B_{2,0}$	$B_{2,1}$	$B_{2,2}$	$\cdots$	$B_{2,N}$	Cost of Trial 2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Trial M	$B_{M,0}$	$B_{M,1}$	$B_{M,2}$	$\cdots$	$B_{M,N}$	Cost of Trial M

Detailed working of the algorithm - First Epoch

Typical first Epoch

Probability Vector	0.5	0.5	0.5	$\cdots$	0.5	Cost of the Trial
Trial 0	0	1	1	$\cdots$	0	42
Trial 1	1	1	0	$\cdots$	1	96
Trial 2	0	1	0	$\cdots$	1	223
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Trial M	1	1	1	$\cdots$	0	31

Detailed working of the algorithm - Some time during the iterations

Say iteration 45 or something else

Probability Vector	0.8	0.1	0.25	$\cdots$	0.75	Cost of the Trial
Trial 0	1	1	1	$\cdots$	0	49
Trial 1	0	0	0	$\cdots$	1	91
Trial 2	1	1	0	$\cdots$	0	145
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Trial M	0	1	1	$\cdots$	0	200

Detailed working of the algorithm - Final iteration

Final iteration - everything has now settled down

Probability Vector	1.0	0.0	0.0	$\cdots$	1.0	Cost of the Trial
Trial 0	1	0	0	$\cdots$	1	25
Trial 1	1	0	0	$\cdots$	1	25
Trial 2	1	0	0	$\cdots$	1	25
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Trial M	1	0	0	$\cdots$	1	25

Submitting via Dropbox

1. Your work must be submitted via the Dropbox link provided
2. Your document must include at least the following parts
    1. A description of your simulation
    2. A description of your software development process
    3. A description of the "logic" of the program. (Flowcharts etc.)
    4. A collection and analysis of your results.
    5. A listing of your code - well commented.

Assessment

Assessment will be according to the following criteria:

- 50% for the "descriptions" listed above.
- 30% for the presentation document you produce.
- 20% for your code listing, with comments.