Programmeren (FEB22012)

7. Exercise

Deadline for submission: 2012-10-19 23:59 CET

Introduction

In this exercise we determine the optimal payload of a cargo freighter with a capacity of 1500 m^3. The freighter will be filled with a set of packages that each have a certain pay-off and volume assigned to them. There are more packages to transport and therefore we have to select which ones to load into the freighter by maximizing the aggregate pay-off while being constrained by the size of the cargo bay.

A simple way to choose an optimal set of packages is to examine all combinations, but as you know from the combinatorial optimization course, this is the knapsack problem that is of exponential complexity with respect to the amount of packages. Therefore we will implement a meta-heuristic called Genetic Algorithm (GA) (see `http://en.wikipedia.org/wiki/Genetic_algorithm`) to approximate the optimal set of packages. GAs are search heuristics that simulate an evolutionary process. The main idea in our GA application is to iteratively generate generations that consists of a set of possible solutions (payloads). The new generation is constructed by mutating (randomly changing the composition) and combining solutions (payloads) of the previous generation. The goal is to eventually find a good estimate of the optimal load. However, bear in mind that this algorithm is a heuristic and does not give any guarantee of optimality of the solution.

Exercise: Genetic Algorithm

You should first implement the genetic algorithm as given in Algorithm 1. The best way to represent the different loads (generations) is as bit-strings, that can be stored in matlab as a matrix row of 0's and 1's. The GA applies a mutation to a selection of loads. Write a mutation function that select a random package and adds it to the load if it wasn't in the load already, or removes it if it was. Write a function that combines two loads into a new one by randomly choosing, for each package, the 0/1 from one of the two original loads. To select loads from the population with a certain probability, you can use the function `randsample`. The GA often finds non-feasible loads, that then stay in the population for some time. Write a function that makes a non-feasible load into a reasonably good feasible load.

Exercise: application

The captain of the freighter can choose from `nPak` packages. The volumes and expected returns of these packages are given in the file `http://smaa.fi/tommi/courses/prog2/data/load.xls` - download this file somewhere where it is accessible by Matlab.

Make a main script that loads the data, generates $n = 60$ random initial loads, and runs the genetic algorithm. The GA will probably perform better if the initial loads represent approximately full ships. You could use the function `randperm` for this. Write a function that calculates the returns of a load, and that assigns a return of 0 to non-feasible loads (loads of higher size than that of the cargo ship).

Create a graph that shows the progress of the GA. This graph should contain two coordinate systems (see function `subplot`). The first coordinate system will be a scatterplot of each loading in each generation (horizontal the generation number and vertical the profits). The second coordinate system represents the best loading in time. The horizontal axis represents time and the vertical axis the package number. When the package is selected in the best loading in generation t, there will appear a point, else not. Give each coordinate system an appropriate title. Show in one of the titles the generation and the best profit. See also Figure 1. Take care that the figure will be updated after every generation.

Algorithm 1 The Genetic Algorithm pseudo-code

```
Parameters:
 nGen:       number of generations.
 n:          number of loads in a generation.
 PercComb:   percentage of loads that are combined.
 PercMut:    percentage of loads that are mutated.

Variables:
 Pop:      collection of loads (current generation)
 PopNew:   collection of loads (new generation)
```

Pseudo-code

- fill Pop with an initial population of n loads.
- calculate yields f_i of the loads i in Pop.
- calculate nComb = PercComb*n and nMut = PercMut*n.
- repeat nGen times:
  - PopNew = empty.
  - Add n-nComb loads from Pop to PopNew; the probability of selecting a load i is

$$\Pr(i) = \frac{\exp(f_i - f_{\max})}{\sum_{j=1}^{p} \exp(f_j - f_{\max})}$$

    with $f_{\max} = \max_i f_i$.
  - Add nComb loads to PopNew by combining randomly selected loads from Pop (with probability $\Pr(i)$).
  - Choose PercMut percent of the loads from PopNew and mutate them.
  - Pop = PopNew.
  - Calculate the yield f_i of loads i in Pop.
  - Store the current best found total load.
- Return best total load.

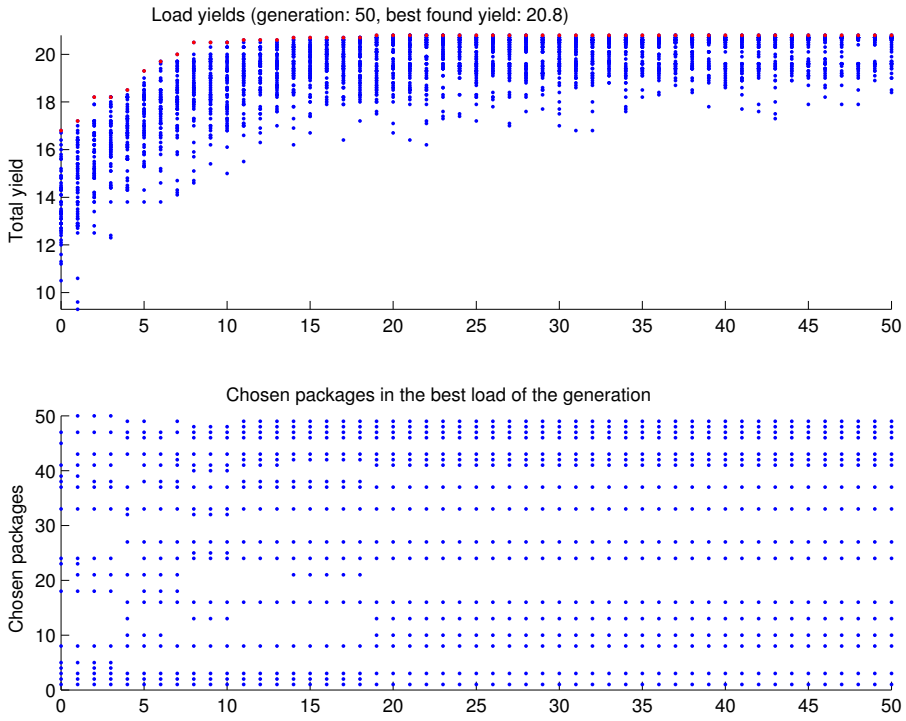

Figure 1: Example of the graph.