Computação Evolucionária

2013/2014

Notes for the PL 3

Ernesto J. F. Costa

25 de Fevereiro de 2014

Class PL 3

Single-State Stochastic Algorithms Revisited

3.1 Algorithms

We presented a set of six algorithms based on the idea of iteratively improving one candidate solution, relying on a stochastic perturbation of the solutions:

- Random Search
- Hill-Climbing
- Hill-Climbing with Random Restart
- Simulated Annealing
- Tabu Search
- Iterated Local Search

In a previous class we start doing some experiments to get the flavor of the functioning of those algorithms. Now it is time to continue this task but now focus more in the design process and in the analysis of the importance of the choice of the different parameters used by the algorithms. For a given problem, the choice of the right algorithm to use is not an easy one and can be tricky. The important point to remember is to clearly understand in what way these algorithms are different and what are their commonalities. As a picture is worth one thousand words let's present the underlying abstract philosophy of all the algorithms (see figure 3.1).

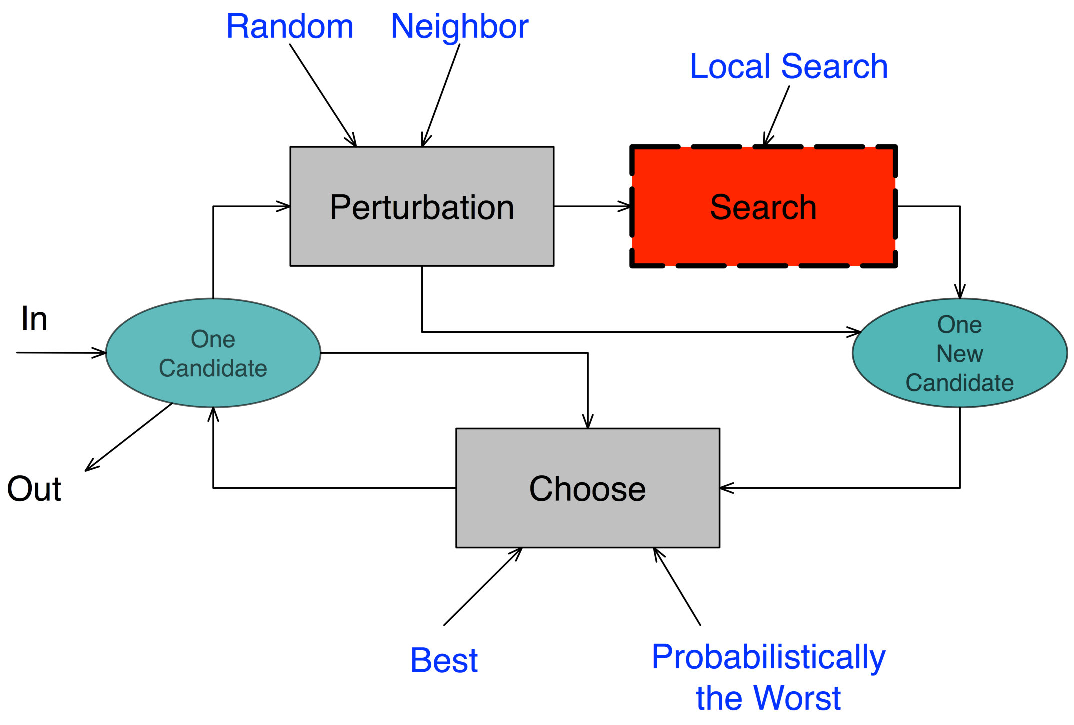

Figura 3.1: Single-State Algorithms: a common view

We can do a random perturbation or chose as next candidate a neighbor of the current one. Also, when updating the candidate we can do this deterministically, keeping always the best, or stochastically, in situations when we sometimes keep the worst of the two candidates. Moreover, we may have a (local) search procedure that try to improve the new candidate before we compare it with the current one.

3.2 Design

It is clear that one of the most important choices as to do with the representation of the problem at hand so we can use one of the algorithms. For the time being we already encounter several tips of representations, namely:

- Binary Strings
- Vector of Floats

- Permutation of Integers

Those representations may implicitly encode some knowledge that we have about the problem. Also, we should choose a representation that is **complete** for the task, i.e., that can encode every possible solution. Additionally, the chosen representation should make easy the task of fabricating the new candidates by perturbation. Last but not the least, the fitness function used to evaluate the candidate solutions should be computable in a convenient time.

It is clear that when we face the engineering problem of designing a good solution all the mentioned points influence each other and some times we face a trade-off, for improving one aspect/component may hinder obtaining a good option for another one. There is sone **Art** involved in the design process!

3.3 The problem: 0/1 Knapsack Problem

3.3.1 Description

The well-known single-objective 0/1 knapsack problem is defined as follows: a thief has a knapsack of a certain **capacity**. He looks around and sees different items of different **weights** and **values**. He would like the items he chooses for his knapsack to have the greatest value possible yet still fit in his knapsack. The problem is called the **0-1 Knapsack Problem** because the thief can either take (1) or leave (0) each item. The knapsack problem is an example of an integer linear problem which has NP-hard complexity.

The problem can be defined more formally as follows. Given a set of items (n, with $i = 1, ..., n$)), each with a weight w_i and a value v_i, the goal is to determine

$$max(\sum_{i=1}^{n} v_i \times x_i)$$

subject to:

$$\sum_{i=1}^{n} w_i \times x_i \leq C$$

with

$$x_i \in \{0, 1\}$$

.

3.4 Choosing the data

The hardness of the problem is sensitive to the choice of the data used (value, weight and maximum value). The weights and the value of each item are obtaining using uniform distributions in a certain interval. Basically, there are three ways to generate the data:

Uncorrelated

P[i] = uniform([1..v])
V[i] = uniform([1..v])

Weakly Correlated

P[i] = uniform([1..v])
V[i] = P[i] + uniform([-r..r])

Strongly Correlated

P[i] = uniform([1..v])
V[i] = P[i] + r

You should be aware that we do not allow negative values, i.e., you have to guarantee that $V[i] > 0$. Also, the more correlated the data the harder the problem.

3.5 Parâmetros

To fabricate the data we need to define concrete values for certain parameters. Typical values are:

- $v = 10$
- $r = 5$

- Número de itens $n \in \{100, 250, 500\}$

Finally, the capacity of the knapsack can be calculated in two different ways:

a) Restrictive: $C = 2 \times v$

b) Average: $C = 0.5 \times \sum_{i=1}^{n} P[i]$

3.6 Goal

The goals of this assignment are:

- Implement the functions that generate the different set of data.
- Choose a representation for the candidate solutions
- Define a fitness function
- Choose an appropriate algorithm
- For the chosen algorithm and representation, define the function that perturb the solutions
- Run the algorithm for different set of data (uncorrelated, weekly correlated, strongly correlated), and number of items (100,250,500), and analyze the results.