Evolutionary Computation

2016/2017

Project I
Traveling Thief Problem

Ernesto J. F. Costa

April 3, 2017

1.1 Introduction

We know that evolutionary algorithms are the option to solve optimization problems either when there is no analytical solution or when they are computationally intractable. In these situations a heuristic approach is the best we can use, paying eventually the price of finding only a near-optimal solution.

The No Free Lunch Theorem states that any two algorithms are equivalent when their performance is averaged across all possible problems [5]. This a general statement but do not preclude the fact that we may find an algorithm that is statistically the best for a particular class of problems or for specific instances of a certain problem. Usually we rely on benchmark problems to test our algorithms. We already met some examples, e.g., different function optimization problems, the João Brandão's Numbers Problem, the Traveling Salesman Problem (TSP), or the 0/1 Knapsack Problem, to name just a few. Ideally these problems must reflect characteristics of real world hard problems.

Optimization problems are difficult to solve due in part to the nature of the fitness landscape of the problem itself (ruggedness, deceptiveness) or because of limitations of the algorithm (premature convergence, robustness). Moreover, the size of the problem, the presence of noise and uncertainty or the existence of constraints are complications added by the real world. That said, the traditional benchmark problems have helped in the task of developming good optimizers.

Over the years there are at least two aspects of real world problems that have been neglected when proposing benchmark problems: **combination**, i.e., the fact they very often are the combination of two or more sub-problems, and **interdependence**, i.e., when the solution of one of the sub-problem influence the quality of the solution of the other sub-problems. To overcome this limitation a new benchmark problem was recently proposed, called **Traveling Thief Problem** (TTP) ([1], [4], [2], [3]). The TTP is a combination of the TSP and the 0/1 KP.

The goal of this project is for you to implement and test an algorithmic solution for the TTP mainly based on evolutionary algorithms (EA). We say mainly because you may use **ideas** from the other single-state algorithms (e.g., local search, tabu search, simulated annealing) discussed in the beginning of course to eventually empower your standard EA. You may use all information available, including code, to facilitate your task. The site `https:`

//sites.google.com/site/mohammadrezabonyadi/standarddatabases/travelling-thief- is the best place to start your search. There you will find links for papers, data sets, code (Java and Matlab mostly). Also, in **InforEstudante** we you will find some material. Beware that even if you can use whatever material you want from others, all original sources must be explicitly be described, and that **originality** will result in bonus points for the final mark.

1.2 Problems Definition

As we said TTP is a combination of two well-known problems: TSP and KP.

Traveling Salesman Problem The TSP involves n cities that a salesman must visit exactly once, minimizing a certain aspect (length of the path, time of the trip). To solve it we must know the localization of the cities or a matrix with the distance between every pair of cities. When time is the criterion we need also to know the velocity of the salesman. More formally, we want to find a tour $\overline{x} = (x_1, \ldots, x_n)$ that minimizes the function:

$$f(\overline{x}) = t_{x_n x_1} + \left(\sum_{i=1}^{n-1} (t_{x_i x_{i+1}}) \right)$$

where $t_{x_i x_{i+1}}$ is the time for the travel between the city x_i and the city x_{i+1}. It is define as a function of the distance (d_{ij}) and the velocity (v_c):

$$t_{x_i x_{i+1}} = \frac{d_{x_i x_{i+1}}}{v_c}$$

Notice that if the velocity is constant then it is as we were just minimizing the distance.

0/1 Knapsack Problem The KP involves a set of m items $I_1, \ldots, I_m$ each one with a certain value (p_i) and weight (w_i), that a thief want to put in its knapsack of limited capacity W. The goal is to maximize the value without exceeding the knapsack capacity. More formally, we want to maximize:

$$g(\overline{y}) = \sum_{i=1}^{m} p_i y_i, \; y_i \in [0, 1]$$

subject to:

$$\sum_{i=1}^{m} w_i y_i \leq W$$

y_i is 0 if the item I_i is not in the knapsack, 1 otherwise.

Traveling Thief Problem The TTP is a combination of the TSP and KP. The thief has to visit exactly once each city, like in the TSP, and when he visit a city he may decide to pick (or not) an item to put in his/her knapsack, like in the KP. A solution for the TTP is a tour $\overline{x}$ and a picking plan $\overline{z} = (z_1, \ldots, z_m)$, $z_i \in \{0 \cup A_i\}$. A_i is the set of cities where the item I_i is available. If $z_i = 0$ that means that no item was picked. It is clear that a good solution for the problem will be one that minimizes the time of the trip and maximizes the value of the items in the knapsack.

Now is time for us to describe the way the two sub-problems may influence each other. We will follow the two models proposed in [1].

Model TTP1 In this model we want to maximize:

$$G(\overline{x}, \overline{z}) = g(\overline{z}) - R \times f(\overline{x}, \overline{z})$$

where $g(\overline{z})$ is the total value for the picking plan $\overline{z}$, as it would be computed by the standalone 0/1 KP, R is the rent per time unit that the thief as to pay while doing the tour, and f is the total time of the tour. The fact that the time depend on the tour and on the picking plan is because the velocity decreases as a function of the weight of the knapsack. In fact the current velocity is given by:

$$v_c = \left(v_{max} - W_c \times \frac{v_{max} - v_{min}}{W} \right)$$

From the formula it is clear that when the knapsack is empty the velocity is maximum and when the knapsack is full the velocity is minimum. On the other hand the rent rate (R) per time unit links the time to the value.

Model TTP2 In this model the velocity is also defined as a function of the weight, but also the value of an item decreases over time. The dropping rate is defined by:

$$Dr^{\lfloor \frac{T_i}{C} \rfloor}$$

where T_i is the total time the item I_i is carried in the knapsack and C is a constant. Now the goal can be stated as follows:

$$G(\overline{x}, \overline{z}) = \begin{cases} \min f(\overline{x}, \overline{z}) \\ \max g(\overline{x}, \overline{z}) \end{cases} \tag{1.1}$$

It must be clear how the sub-problems influence each other. The more items we pick the more value we get, but the speed decreases and more time means that the final value of each item decrease.

1.3 Work to do

You should start by reading the bibliography. In particular, you must read carefully the paper [1]. Then choose one model (TTP1 or TTP2), and the corresponding parameters, and focus on the way the **fitness** of each solution is calculated. Then, devise an evolutionary algorithm for the TTP, defining the **representation** and the corresponding **variation operators**. Finally, choose the other components of the EA (e.g., selection mechanisms (parents and survivors, initial population), and parameters (e.g., population size, probability of application of the variation operators). Due to the stochastic nature of the optimizer **run your algorithm 30 times** and store the results of the best over all over generations and the average best over generations. Do this for different data sets (at least 3). Visualize the results. You will find some data sets following the link provided above or in http://cs.adelaide.edu.au/~optlog/CEC2015Comp/. In that web page you will find also the best known value for each of the problems, so you can compare the quality with that of your own results.

1.4 Rules to follow

The work is to be done by a group of up to four students. All elements of the group are supposed to contribute evenly to the work. You may consult and use freely whatever sources you want, but the group is supposed to keep all discussions and clarifications within the group. The groups will compete among them for the best results, and **the originality and quality of the results will be rewarded**!

Moreover, the code and the data, you are supposed to deliver a **document** written as if it was a paper for a conference describing the approach and a **short user manual** so I can use it myself. The document should not have more than 15 pages. A template describing the form of the document

will be provided. We advise the use of LaTeX as word processor, but you may used the word processor of your preference.

Deadline The work (document, code, user manual, data sets used) should be delivered no later than **14th of May**. This a hard deadline!

Bibliography

[1] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. *IEEE Congress on Evolutionary Computation*, pages 1037–1044, 2013.

[2] Mohammad Reza Bonyadi, Zbigniew Michalewicz, Michao Roman Przybyoek, and Adam Wierzbicki. Socially inspired algorithms for the travelling thief problem. *GECCO*, pages 421–428, 2014.

[3] Hayden Faulkner, Sergey Polyakovskiy, Tom Schultz, and Markus Wagner. *Approximate Approaches to the Traveling Thief Problem*. ACM, New York, New York, USA, July 2015.

[4] S Polyakovskiy, M R Bonyadi, and M Wagner. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the ...*, 2014.

[5] D H Wolpert and W G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.