Advanced control class

Ecole Centrale Paris

Exercise Solution: Pure random search and (1+1)-EA on $\Omega = \{0,1\}^n$

January 24, 2014

Anne Auger, Dimo Brockhoff
anne.auger@inria.fr
dimo.brockhoff@inria.fr
http://researchers.lille.inria.fr/˜brockhof/advancedcontrol/

Pure Random Search (PRS)

The first stochastic optimization algorithm, introduced before any genetic algorithm (GA) or evolution strategy (ES), is the so-called pure random search, or PRS for short[1]. Assuming a bounded search domain, the algorithm consists in sampling independently points *uniformly* distributed in the search space.

We will implement the algorithm PRS in the context of the maximization of functions defined on the space of bit strings of a certain length n, i.e. on the search space $\Omega = \{0,1\}^n$.

Pure Random Search to maximize $f : \{0,1\}^n \to \mathbb{R}$

```
Initialize uniformly at random $\boldsymbol{x} \in \Omega = \{0,1\}^n$
while not terminate
    Sample $\boldsymbol{x}'$ uniformly at random in $\Omega$
    if $f(\boldsymbol{x}') \geq f(\boldsymbol{x})$
        $\boldsymbol{x} = \boldsymbol{x}'$
return $\boldsymbol{x}$
```

Example 1 *For $\boldsymbol{x} \in \{0,1\}^n$, the function* ONEMAX *is defined as*

$$f_{\textsc{OneMax}}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{x}_i$$

1. What is the maximum of the ONEMAX function? What is the value of $f_{\textsc{OneMax}}$ at the optimum?
   The maximum of the ONEMAX function is the all-one string $(1, \dots, 1) = 1^n$ and its value is $f_{\textsc{OneMax}}(1^n) = n$.

[1] The historical papers proposing the use of the PRS are:

- ⋆ S.H. Brooks: "Discussion of random methods for locating surface maxima". Operations Research 6 (1958), pp. 244–251.
- ⋆ L.A. Rastrigin: "The convergence of the random search method in the extremal control of a many-parameter system". Automation and Remote Control 24 (1963), pp. 1337–1342.

2. Write a Matlab function `onemax.m` that takes as argument a bitstring $\boldsymbol{x}$ of arbitrary length and gives back the value $f_{\textsc{Onemax}}(\boldsymbol{x})$ (useful Matlab instruction: `sum`).
   Though the function itself is very simple and writing it in a separate function file seems to be superfluous, it is good practice in terms of object-oriented programming and in a later improved implementation, the fitness function can be easily replaced by more complicated problems. The MATLAB code can be found in the file `prsAndOnePlusOne.zip` with name `onemax.m`.

3. Write a Matlab function `pureRS.m` that takes as argument the search space size n and returns the number of function evaluations needed to reach the optimum of the ONEMAX function, as well as a vector `fitness` that contains the sequence of the best-ever objective function values found so far. In other words, the i^{th} coordinate of the vector `fitness` contains the best function value found until iteration i. To sample, you might use the Matlab instructions `rand` and `round`.
   This function can be found as well in the archive `prsAndOnePlusOne.zip`.

4. Plot the evolution of the function value ("fitness") as a function of the number of function evaluations for two independent runs of the algorithm. What do you observe? You might want to use the instructions `plot` and `hold on`.
   The script `evolOfFitness.m` in `prsAndOnePlusOne.zip` shows how to plot the fitness values over time by running the above function for the PRS. We observe a high variance of the results among independent runs of the algorithm. This is the reason why in the next step, we average over 11 runs of the algorithm and later on investigate the *expected* runtime.

5. Write a script `RunningTime.m` to plot for $n = 1{:}2{:}14$ the empirical expected value of the time needed to reach the optimum of $f_{\textsc{Onemax}}$ with the PRS algorithm. Plot as well the standard deviation around the expected value. We advise to compute the empirical expected value and the standard deviation based on 11 independent runs of the algorithm.
   This script can also be found in `prsAndOnePlusOne.zip`.

Note: the time to reach the optimum is measured by counting the number of function evaluations needed to reach the optimum and not the real wall-clock time. Indeed, internal operations are generally negligible compared to the cost of evaluating the objective function.

6. Compute theoretically the expected time to reach the optimum as a function of n. Compare the theoretical and empirical results. Hint: show that the time to reach the optimum follows a geometric distribution with a parameter to determine.
   In order to compute theoretically the expected time to reach the optimum for the PRS, we introduce the notation T_{opt} for the number of times, a certain run of PRS queries the objective function (keyword: "black box") until reaching the search point 1^n. Note that T_{opt} is a random variable and we are interested in computing its expectation $E(T_{\text{opt}})$.
   We observe that the probability that PRS is reaching the optimum in the next step is independent of its current search point as well as independent from any other event in the past. This probability to reach the optimum is always $p = 1/2^n$ due to the fact that only one out of the total 2^n search points is optimum and PRS is sampling a new search point uniformly at random in each iteration. Because of the independency of each step, the search for the optimum is a Bernoulli experiment and T_{opt} geometrically distributed[2]. Hence, the expectation of T_{opt} is $E(T_{\text{opt}}) = 1/p = 2^n$ (and its variance $\frac{1-p}{p^2}$).

(1+1)-EA

We will now implement a simple evolutionary algorithm, the so-called (1+1)-EA. Its population is reduced to a single individual. The single parent (the first "1" in the notation (1+1)) mutates to give an offspring (the second "1" in the notation (1+1)). The best among the offspring *and* the parent is kept for the next iteration (symbolized by the "+" in the notation (1+1)).

[2]See for example `http://en.wikipedia.org/wiki/Geometric_distribution` for more details.

The mutation used is the so-called bit-flip mutation where each bit of the parent is changed with probability $1/n$ (and thus stays unchanged with a probability $1 - 1/n$).

(1+1)-EA to maximize $f : \{0,1\}^n \to \mathbb{R}$

```
Initialize uniformly at random $\boldsymbol{x} \in \Omega = \{0,1\}^n$
while not terminate
    Create $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with probability $1/n$
    if $f(\boldsymbol{x}') \geq f(\boldsymbol{x})$
        $\boldsymbol{x} = \boldsymbol{x}'$
return $\boldsymbol{x}$
```

7. Follow again the questions 3, 4, and 5 for the (1+1)-EA. The corresponding code can be also found in the provided `prsAndOnePlusOne.zip` file.

8. The theoretical complexity[3] of the expected time to reach the optimum is $\Theta(n \log n)$ for the (1+1)-EA on OneMax. Compare the theoretical and empirical results. Give an idea for the theoretical proof of the upper bound of $cn \log n$. The experimental comparison can be found in the file `RunningTime1p1EA.m` while for the proof of the expected runtime, we refer to the provided slides.

9. Explain the differences obtained between the PRS and the (1+1)-EA. The differences between the two algorithms are caused by the different ways to produce a new solution. While for PRS the probability to reach each solution stays the same over time, these probabilities depend on what happened in the past for the (1+1)-EA. More precisely, the current probability distribution on the search space depends on the current search point $\boldsymbol{x}$ and puts more probability mass to points close to the current search point than to other points further away. For the OneMax function, this is a good strategy because the neighborhood of good points contain a larger amount of equally or better search points than for points with smaller function value. This reduces the runtime from exponential (for PRS) to polynomial (for the (1+1)-EA) in n and, consequently, from impractically high to fast.

Another Function

We consider now the function `Needle in the Haystack` defined as follows.

Example 2 *For $\boldsymbol{x} \in \{0,1\}^n$, the function* Needle *is defined as*

$$f_{\text{Needle}}(x) = \begin{cases} 1 & \text{if } \boldsymbol{x} = (1, \ldots, 1) \\ 0 & \text{otherwise} \end{cases}$$

10. What will be the performances of PRS and (1+1)-EA on the function f_{Needle}? Comment on the differences w.r.t. the function f_{OneMax}.

This part has not been part of the exercise this year.

[3]Reminder: $f(n) \in \Theta(g(n))$ if $\exists k_1, k_2,\ |g(n)| \cdot k_1 \leq |f(n)| \leq |g(n)| \cdot k_2$.