max planck institut informatik

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Benjamin Doerr **Winter 2012/13**

Exercises for Randomized Methods in Computer Science

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws12/rmcs/

Assignment 6 Due: Wednesday, November 28, 2012

Exercise 1 Understand the "quasirandom with restarts" rumor spreading protocol that was introduced in the last lecture. To this aim, read the ICALP 2011 paper of Doerr and Fouz (will be on the course page). Read it completely, take cursory note of the very precise bounds proven in the paper and of the other works mentioned, but gain a deeper understanding of the working principles of this protocol. Answer the following questions each in few sentences (in particular, without proofs).

a) Why is the basic version of this protocol not robust against adversarial failures? How is this overcome? Could this also be overcome with the following trick: After the adversary failed some nodes, the remaining nodes choose a random permutation and use this order instead of the natural order 1, 2, 3, ... When using a random order, an adversarial choice of failed nodes should be equivalent to a random choice.

b) Why (on a superficial level) does this protocol not work on a random graph of n vertices (n nodes with each possible edge present with probability $1/2$)?

c) Comment on the following alternative protocol: When a node i is informed, it next calls node $i+1$ (modulo n, of course), then it calls random nodes. After R times having called an informed node, it stops.

Exercise 2 In the next lecture, we shall start with the analysis of elementary randomized search heuristics. As a warm-up, consider the following problems. Our aim is always to find the maximum of a function $f : \{0,1\}^n \to \mathbb{R}$. The elements of $\{0,1\}^n$ naturally are called bit-strings. We shall regard the following two randomized search heuristics.

Randomized local search (RLS) starts with a random bit-string and repeatedly tries to improve it by flipping a randomly chosen bit. In the following pseudo-code, no termination criterion is specified. This is because we shall be interested in questions like "does this approach eventually find the optimal solution" and "how long does this take". In practice, of course, one always has some termination criterion like a fixed number of iterations or a bound how many iterations without improvement are tolerated.

Algorithm 1: Randomized Local Search (RLS)

Choose $x \in \{0,1\}^n$ uniformly at random;
repeat
  $y := x$;
  Choose $i \in [n]$ uniformly at random;
  $y_i := 1 - y_i$; `// flip the ith bit`
  if $f(y) \geq f(x)$ **then**
    $x := y$
until *forever*;

A slight variant of RLS is what is called the *(1+1) evolutionary algorithm ((1+1) EA)*. The only difference is that now y is not obtained from flipping one random bit, but from flipping each bit independently with probability $1/n$.

Algorithm 2: (1+1) Evolutionary Algorithm ((1+1) EA)

Choose $x \in \{0,1\}^n$ uniformly at random;
repeat
  $y := x$;
  for *each* $i \in [n]$ **do**
    with probability $1/n$ do $y_i := 1 - y_i$
  if $f(y) \geq f(x)$ **then**
    $x := y$
until *forever*;

a) Convergence: A first useful property of a randomized search heuristic is convergence, that is, that is surely finds an optimum (=maximum) of f if one waits sufficiently long. For which functions $f : \{0,1\}^n \to \mathbb{R}$ do RLS and the (1+1) EA converge? Note: I'm really only asking for convergence, so I don't care how fast this convergence is (in this part of this exercise).

b) Run-time analysis: Indeed, it would be nice to know how long it takes (in expectation or with high probability) until an optimal solution is found. An easy test function for randomized search heuristics is the so-called ONEMAX function, simply counting the number of 1-bits and formally defined by $f : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^{n} x_i$. Prove (formally) that both RLS and the (1+1) EA find the optimum of ONEMAX in time $\Theta(n \log n)$ and not faster. By time we here always mean the number of evaluations of the function f. Think (informally) about what happens if we replace ONEMAX with a general linear function $f : \{0,1\}^n \to \mathbb{R}; x \mapsto \sum_{i=1}^{n} a_i x_i$, where the a_i are arbitrary positive numbers. Can you still prove an $O(n \log n)$ upper bound?

Exercise 3 This is my favorite example of dependent randomization (well, maybe you only see dependencies and no randomization, that's OK).

Imagine that I organize the final exam as follows: I put a hat on each of your heads. This hat is either blue or red. Each of you can see all hats except his/her own. The sole exam problem is to find out your own hat color. As usual, this is a written exam and you can't talk to your colleagues. Since this exam is easy to mark, I do this immediately after the exam. Those who correctly found their hat color will pass, those who don't guess right will fail and are immediately sent to the secret dungeon of unsuccessful students in the MPI basement. Faced with this situation, you should find a strategy that ensures that at least one student gets the hat color correct, so that he/she can run to the dean and complain about the unfair exam.

So the task in this problem is finding a strategy that ensures that one student guesses right, or proving that no such strategy exists. The students may agree on a strategy before entering the exam room, but from then on no communication of any kind is possible (not a single bit of information will flow from any student to any other).

Note that guessing random colors indeed has a very high chance of getting at least one correct answer, but this is not what this problem asks for. We (that is, you) want a guarantee that there is at least one correct answer.