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 10 Due: Friday, January 11, 2013

Exercise 1 EA for combinatorial problems. Given an undirected graph $G = (V,E)$, a Eulerian cycle is a tour (or closed *walk*) through the graph that uses each edge exactly once. Discuss possible representations and mutation operators for evolutionary algorithms to compute a Eulerian cycle. What optimization times do you envisage for the (1+1) EA using these setups?

Exercise 2 Crossover: Get an overview of the main results of the paper

Thomas Jansen, Ingo Wegener: The Analysis of Evolutionary Algorithms - A Proof That Crossover Really Can Help. Algorithmica 34(1): 47-66 (2002)

In particular, understand what example they use to demonstrate the usefulness of crossover. What are the strengths and weaknesses of this result?

Exercise 3 Mastermind (Weihnachtsaufgabe): In the first lecture, we regarded the black-peg Mastermind game with n positions and $k = n$ colors (black-peg: you only learn in how many position guess and secret code agree). We proved that random guessing finds the secret code in $O(n \log n)$ rounds. One can prove formally that random guessing cannot do better by making the following informal, but useful, consideration precise: A random guess typically has at most a constant number of positions correct. Hence a constant number of bits (on average) suffice to encode each answer. Since there are n^n secret codes, we need $\Omega(n \log n)$ bits to encode the secret (no matter how clever we encode). A set of answers determining the code is also a way of encoding, hence we need at least $\Omega(n \log n)$ guesses.

In this exercise, we will find better strategies than random guessing. The main challenge, obviously, will be asking guesses that reveal more than a constant amount of information. These, necessarily, cannot be random and independent from the history of the game. This seems difficult at first—note that if I choose the secret randomly, any first guess you do has an expected answer of one.

The good news is that there is a way to learn from previous guesses: If the answer to guess $(g_1, \dots, g_n)$ is zero, then we know that color g_i cannot occur on position i. Consequently, we reduced the number of relevant colors (for each position) to $n-1$. We will build on this idea in the following.

To this aim, note (and use whereever helpful) the following extension of the random guessing re-

sult: for all $k \leq n$, one can find the secret code in an n position k-color Mastermind game with $O(n \log k / \log(n/k))$ random guesses. Here it does not make a difference if we really play with k colors only, or if we have potentially different sets of k colors for each position.

a) [easy] Analyse and optimize (find the right value for K) the following strategy: (i) While the number of colors possible at each position is greater than K, ask a random guess composed of possible colors. If the answer is zero, this reduces the number of colors possible at each position by one. After this, use the above result on random guessing with K colors to finish the game.

b) [easy] Analyse and optimize the following strategy: Repeat asking random queries (composed of all colors) until the zero-answers obtained reduce the number of possible colors at each position to K. Then use the above result on random guessing with K colors...

c) [difficult] What is the weakness of the strategy in (b)? Can you find a better strategy? Anything that gives more than $O(n\sqrt{\log n})$ is interesting. $O(n \log \log n)$ is the best I know.

d) [immediate PhD] Close (or reduce) the gap between the trivial lower bound of $\Omega(n)$ and $O(n \log \log n)$.