TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Exam IN4301 Advanced Algorithms

January 24 2011

- Use a separate sheet for each question.
- This is a closed book examination with 4 questions worth of 100 points in total.
- Your mark for this exam will be the number of points divided by 10.
- If you have at least a 5.0 for this exam, the average of the two programming exercises, as well as the average of the 8 homework exercises, your final mark for this course is the average of these three marks, rounded to the nearest half of a whole number. That is, 9.7 is rounded to 9.5, and 5.8 is rounded to 6.
- Use of book, readers, notes, and slides is not allowed.
- Use of (graphical) calculators is not permitted.
- Specify your name, student number and degree program, and indicate the total number of submitted pages on the first page.
- Write clearly, use correct English, and avoid verbose explanations. Giving irrelevant information may lead to a reduction in your score.
  Notice that almost all questions can be answered in a few lines!
- This exam covers Chapters 10 and 11 of Kleinberg, J. and Tardos, E. (2005), *Algorithm Design*, all information on the slides of the course, and the set of papers as described in the study guide.
- The total number of pages of this exam is 5 (excluding this front page).

1. (a) (6 points) Suppose that for a given bounded search tree it is known that each node representing a problem of size n has at most two child nodes, one representing a problem of size at most $n-1$ and one of size at most $n-4$. Furthermore suppose that these child nodes can be found in linear time. Describe how you would determine an upper bound on the run time of an algorithm using this search tree on an input of size n.

Solution: First we represent the upper bound on the runtime by a recursive function $T(n) = n + T(n-1) + T(n-4)$ with $T(n) = 1$ in case $n \leq 4$. The solution to this recurrence can be obtained by reformulating $T(n)$ to α^n, so $\alpha^n = \alpha^{n-1} + \alpha^{n-4}$, and then by solving this numerically, which leads to an upper bound of about $O^*(n \cdot 1.4^n)$.

(b) (6 points) Suppose we have a problem instance with a parameter k, and an algorithm for that problem with a run time bounded by $f(k) \cdot p(n)$ time, where

- f is a (usually exponential) function depending only on the parameter k
- p is a polynomial function.

Is this problem fixed parameter tractable? Is it kernelizable?

Solution: Yes, it is FPT by definition. Yes, it is kernelizable because a problem is fixed-parameter tractable if and only if it is kernelizable (known theorem).

For the following two questions, let a graph $G = (V, E)$ with n vertices as well as a tree decomposition $(T, \{V_t\})$ of G be given.

(c) (7 points) Let any set of vertices $S \subseteq V$ be given. Prove that if for every piece $t \in T$ the subgraph induced by $V_t \setminus S$ does not contain any cycle, then G does not contain any cycle. (Hint: Use the property of a tree decomposition T that says that if a piece is removed from the graph, the resulting subgraphs for two different pieces have no edge with one end in each of them.)

Solution: Suppose that no piece in T contains a cycle. Suppose there is a cycle $C = v_1, v_2, \ldots, v_k$ in G. Then C is not contained in a single piece t, i.e., for every t, $C \not\subseteq V_t$. Thus there should be two vertices $v_i, v_{i+1} \in C$ such that $v_i \in V_{t_1}$ and $v_{i+1} \in V_{t_2}$ with $v_i \notin V_{t_2}$ and $v_{i+1} \notin V_{t_1}$ for two pieces $t_1 \neq t_2 \in T$. Choose now any t on the path from t_1 to t_2. However, property 10.13 in the book then says that if this piece t is removed from the graph the resulting subgraphs for two different pieces have no edge with one end in each of them. Clearly, there can be no cycle in G.

(d) (6 points) Consider the problem of finding a subset S of V of minimal size such that the graph induced by $V \setminus S$ does not contain any cycles. Using the statement in the previous question, the following recursive function determines the minimal size of such a set S. Initially this function is called by Min-FVS(T,r,V_r,$\emptyset$) where r is the root node of the tree decomposition T. For a call on a node t, the third argument denotes the vertices in V_t still to be considered, and the fourth argument the vertices in V_t for which a decision has already been made.

$$\text{Min-FVS}(T, t, V, S) = \min_{\{S' \subseteq V \mid \text{no cycle in } G[V_t \setminus (S \cup S')]\}} \left(|S'| + \sum_{\text{child } t' \text{ of } t} \text{Min-FVS}(T, t', V_{t'} \setminus V_t, (S \cup S') \cap V_{t'})\right)$$

Suppose the tree decomposition has width w. Determine a tight upper bound on the run time of this algorithm and explain how you arrive at that bound.

Solution: In the recursion each piece of the tree is visited. There are at most n such pieces. For each piece, there are at most 2^w calls for all possible subsets that have (not) been considered S. In one call all $O(2^w)$ possible subsets S' are considered. The recursive calls to all children are amortized. This results in an upper bound on the run time of $O(n \cdot 4^w)$.

2. (a) (4 points) Give the general definition of the performance ratio of an approximation algorithm A for a problem X.

Solution: see slide 49, lecture 2.

(b) (6 points) In the Subset Sum problem, we have a multi-set S of positive integers and a target sum t. The problem is to collect a subset S' from S such that the sum of all the integers in S' is not larger than t and comes as close to t as possible. Suppose we have an approximation algorithm A for this problem. We apply A on a problem instance where $t = 100$. The algorithm returns a solution S' where the total sum of the integers in S' equals 80. Is it justified to conclude that the approximation ratio of the algorithm A is at least 1.25? Motivate your answer.

Solution: No, that conclusion is not justified since we don't know what the optimal value is. For example, it might be that for every subset S' for which the total sum is not larger than $t = 100$, it holds that the total sum is at most 80. Hence, the result of applying the algorithm A to this instance might be even equal to the optimal outcome.

(c) (8 points) The Longest Processing Time (LPT) algorithm is an approximation algorithm that is an $\frac{4}{3} - \frac{1}{3m}$ approximation algorithm for the load balancing problem, where m is the number of machines. This means that for 2 machines the algorithm achieves a $\frac{7}{6}$-approximation ratio. We can do better if we apply LPT to the following class of instances: There are 2 machines and the set of jobs consists of jobs having a processing time between 0.5 and 5 and the total processing time T of all jobs together is at least 400. Prove that for this class of instances the approximation ratio of LPT is bounded above by $81/80$.

Solution: Notice that the LPT algorithm will always ensure a final result where the absolute difference Δ between the two loads L_1 and L_2 is strictly less than 5. The minimum value opt of the makespan is at least $(L_1+L_2)/2 = T/2$. Hence the performance ratio is bounded above by

$$\frac{T/2 + \Delta/2}{T/2} = \frac{T + \Delta}{T}$$

Since $T \geq 400$ and $\Delta < 5$, it follows that this approximation ratio is bounded above by $405/400 = 81/80$.

(d) (7 points) What is an $[a, b]$-gap introducing reduction from an NP-hard decision problem A to a minimization problem B? Give a definition to answer this question.

Give a simple argument for the proposition that if there exists such an $[a, b]$-gap introducing reduction and B has a b/a-approximation algorithm, A can be solved in polynomial time.

Solution: See slide 23 - 30, lecture 4.

3. (a) (4 points) In a given primal linear program one of the specific constraints turns out to be an equality. Consider the dual variable associated with this constraint. What are its sign restrictions in the dual program?

Solution: In the dual no sign restrictions are imposed.

(b) (8 points) Consider a Boolean linear program, where we choose Boolean variables -1 and 1. Let a, b and c be Boolean variables occurring in the program. Consider the non-linear inequality $ab + bc + ac \geq -1$ (a so called triangle inequality). Show that we may add this inequality to the program without eliminating solutions and show that it becomes a linear inequality in the semi-definite relaxation. There are three other such triangle inequalities of the form $(+/-)ab(+/-)bc(+/-)ac \geq -1$. Which ones?

Solution: All 8 Boolean combinations satisfy this inequality. There are four such triangle inequalities: the ones with an even number of minus signs.

(c) (5 points) In a given integer linear program four of the inequalities are

$$3x - 2y + z \leq 11 \tag{1}$$

$$x + y - 3z \leq 6 \tag{2}$$

$$-x + 2y \leq -1 \tag{3}$$

$$z \leq 4 \tag{4}$$

Derive the Chvatal cuts

$$x - z \leq 4 \tag{5}$$

and

$$y \leq 3. \tag{6}$$

Solution: $1 \times (1) + 2 \times (2)$ geeft de eerste, en $(5) + (3) + (4)$ geeft de tweede.

(d) (8 points) We consider the graph with two points, being connected with an edge of weight 1. The solution for the max-cut problem is trivial: the max-cut value is 1 and the solution consists of just cutting the only existing edge. Write down the semi-definite relaxation for this problem explicitly and solve it by hand. Explain the calculations clearly in terms of the sd relaxation. Hint: a matrix is semi positive definite iff all its eigenvalues are nonnegative.

Solution: The problem comes down to calculate a symmetric 2 by 2 matrix with diagonal elements 1 and unknown off-diagonal say a. Eigenvalues then are $1 + a$ and $1 - a$, thus the sd restriction boils down to $1- \leq a \leq 1$. This gives the solution $a = -1$. The solution matrix therefore has rank one and the problem is completely solved as expected.

4. In the last three lectures, the class of evolutionary algorithms, the subclasses of evolutionary local search and estimation-of-distribution algorithms as well as the (re)design of these algorithms for multi-objective optimization were discussed.

   (a) (6 points) What is a marginal product model, how can it be used in optimization with an estimation-of-distribution algorithm and why is it sometimes important to do so?

   Solution: A marginal product model (MPM) is a product of probability distributions over mutually exclusive vectors of random variables, thereby representing a factorized probability distribution over all random variables contained in the mutually exclusive vectors.

   A MPM can be used in optimization with an EDA by estimating it from the selected solutions and using it to sample new solutions from. To do so, a greedy algorithm can be used to iteratively join two factors that maximize the increase in MDL score, starting from the univariate factorization. To sample, each factor can be sampled independently, drawing an instance of the factor variables with the probability it was estimated with from the selected solutions.

   It can be important to use an MPM if there are significant dependencies inside subsets of problem variables. If these dependencies are not taken into account, the probability of sampling good instances for dependent variables is much smaller, causing the EDA to scale up far worse (typically exponentially versus polynomially).

   (b) (6 points) Explain the difference between (general) evolutionary algorithms and evolutionary (genetic) local search and describe and explain the conditions under which evolutionary local search is generally considered to be superior.

   Solution: In genetic local search a local search operator is applied to every solution at the moment it is generated by the evolutionary operators. In this way, the evolutionary algorithm performs its search in the space of local optima.

   If the fitness can be partially evaluated, local search operators that change solutions partially before evaluating the change in quality require only little time to make (potentially) big improvements. A function evaluation within the local search procedure can then typically be done in constant time rather than linear in the number of variables. This is typically the case for combinatorial optimization problems.

   (c) (6 points) Explain how a single-objective local search algorithm can be made to work via distance-based scalarization if the optimization problem is multi-objective and describe for any solution a way to do distance-based scalarization such that it can be found in the Pareto optimal set.

Solution: Scalarization is a way to combine all objectives into one objective. Hence, this one objective can be optimized directly using existing single-objective optimization methods. Distance-based scalarization is a way of computing the distance (a scalar) to the utopian point $U(\boldsymbol{f}) = (\min_{\boldsymbol{x}}\{f_0(\boldsymbol{x})\}, \min_{\boldsymbol{x}}\{f_1(\boldsymbol{x})\}, \ldots \min_{\boldsymbol{x}}\{f_{m-1}(\boldsymbol{x})\})$ of a multi-objective optimization problem.

By taking the distance to be the weighted Tchebycheff distance,

$$\textit{arg}\ \min_{\boldsymbol{x}} \left\{\max_{i\in\{0,1,\ldots,m-1\}} \{w_i|U(\boldsymbol{f})_i - \boldsymbol{f}_i(\boldsymbol{x})|\}\right\},$$

for every Pareto-optimal point $\boldsymbol{x}$ there is some configuration of the weights under which $\boldsymbol{x}$ is optimal.

(d) (7 points) Explain why using a mating restriction (preventing certain solutions to be combined with other solutions in the variation operator) in multi-objective optimization can be beneficial and give an example of how a useful mating restriction for multi-objective optimization can be established.

Solution: Different regions along the Pareto front may be very different. For instance, the far ends of the optimal Pareto front are optimal solutions for individual objectives f_i and may therefore have very little in common. By restricting mating (variation) to solutions "near" each other in objective space, you avoid combining solutions that have a high quality for very different reasons.

One useful way of establishing a mating restriction is by clustering the selected solutions in the objective space and performing variation only with solutions that belong to the same cluster.

End of test