Operations Research, Spring 2022 (110-2)
Homework 2

Ling-Chieh Kung*

April 23, 2022

1 Rules

- This homework is due at ***23:59, April 24***. Those who submit their works late but are late by less than one hour gets 10 points off. Works that are late more than one hour get no point.
- For this homework, students should work individually. While discussions are encouraged, copying is prohibited.
- Please submit a ***PDF file*** through NTU COOL and make sure that the submitted work contains the student ID and name. Those who fail to do these will get 10 points off.
- You are required to ***type*** your work with LaTeX (strongly suggested) or a text processor with a formula editor. Hand-written works are not accepted. You are responsible to make your work professional in mathematical writing by following at least the following rules:[1]
  1. When there is a symbol denoted by an English letter, make it italic. For example, write $a + b = 3$ rather than a + b = 3.
  2. An operator (e.g., +) should not be italic. A function with a well-known name (e.g., log, max and sin) is considered as an operator.
  3. A number should not be italic. For example, it should be $a + b = 3$ rather than $a + b = \mathit{3}$.
  4. Superscripts or subscripts should be put in the right positions. For example, a_1 and $a1$ are completely different: The former is a variable called a_1 while the latter is actually $a \times 1$.
  5. When there is a subtraction, write $-$ rather than -. For example, write $a - b = 3$ rather than a - $b = 3$. The same thing applies to the negation operator. For example, write $a = -3$ rather than a = -3.
  6. If you want to write down the multiplication operator, write $\times$ rather than *.
  7. For an exponent, write it as a superscript rather than using ^. For example, write 10^2 rather than 10^2.
  8. There should be proper space beside a binary operator. For example, it should be $a + b = 3$ rather than a+b=3.

  Those who fail to follow these rules may get at most 10 points off.
- As we may see, there are many students, many problems, but only a few TAs. Therefore, when the TAs grade this homework, it is possible for only some problems to be randomly selected and graded. For all problems, detailed suggested solutions will be provided.

*Department of Information Management, National Taiwan University. E-mail: lckung@ntu.edu.tw.

[1] A more complete list of formatting rules is on NTU COOL.

2 Problems

1. (25 points) Consider the following LP

$$\begin{aligned} \min \quad & x_1 + 2x_2 + 3x_3 \\ \text{s.t.} \quad & x_1 + x_2 \geq 4 \\ & x_1 + x_2 + x_3 \leq 9 \\ & x_3 \geq 3 \\ & x_i \geq 0 \quad \forall i = 1, ..., 3. \end{aligned}$$

   (a) (5 points) Find the standard form of this LP.

   (b) (5 points) List all the basic solutions and basic feasible solutions for the standard form of this LP.

   (c) (5 points) List all the extreme points of the feasible region of the original LP. DO NOT prove that they are extreme points; just list them.

   (d) (10 points) Use the simplex method (with the two-phase implementation, if needed) and the smallest index rule to solve the LP. Write down the complete process, optimal solution to the original LP, and its associated objective value. Is there any iteration that has no improvement?

2. (15 points) Consider the following integer program, which represents an instance of the "multi-copy knapsack problem:"

$$\begin{aligned} \max \quad & 2x_1 + 2x_2 + 5x_3 + 11x_4 + 10x_5 \\ \text{s.t.} \quad & x_1 + 4x_2 + 3x_3 + 5x_4 + 3x_5 \leq 20 \\ & x_i \in \mathbb{Z}_+ \quad \forall i = 1, ..., 5, \end{aligned}$$

   where $\mathbb{Z}_+$ is the set of nonnegative integers.

   (a) (5 points) Use the greedy algorithm introduced in class to solve the linear relaxation of this integer program.

   (b) (10 points) Use the branch-and-bound algorithm to solve the original integer program. Depict the full branch-and-bound tree. Do not write down the solution process of each node; write down just an optimal solution and its objective value of each node.

3. (20 points; 5 points each) There are n jobs to be selected and scheduled on m parallel machines. All machines are identical. For job j, the required processing time is p_j, and the amount of benefit that can be collected upon completing the job is b_j. A job can be processed by exactly one machine. For each machine, let its *total processing time* and *total benefit* be the sums of processing times and benefits of all jobs assigned to it, respectively. The capacity of each machine is K, which is the maximum total processing time that a machine may have. Note that as machines have capacity limits, it is possible that not all jobs may be selected and scheduled. In this case, a subset of jobs should be selected and scheduled.

   (a) Formulate an IP to maximize the benefit earned by a machine earning the *least* benefit among all machines.[2]

   (b) Write a program to invoke a solver (e.g., write a Python program to invoke Gurobi Optimizer) to solve the *linear relaxation* of following instance: $m = 3$, $n = 10$, $b = (5, 8, 4, 6, 3, 7, 6, 9, 5, 8)$, $p = (3, 6, 5, 4, 1, 8, 5, 12, 7, 6)$, and $K = 15$. Note that b_js and p_js are expressed in vectors, where the jth element of b and p are b_j and p_j, respectively. Write down the objective value of the optimal solution you obtain. Is it a lower bound or upper bound of the objective value of an IP-optimal solution? Why? Finally, copy and paste your computer program onto your report.

[2]This problem can be proved to be NP-hard.

(c) Consider the following heuristic algorithm HBF (highest benefit first) for solving the problem:

i. First sort jobs according to their benefits b_js and break ties by putting jobs with smaller indices in front of those with larger indices.

ii. Try to schedule each job one by one. When attempting to schedule a job, try to schedule it to the machine currently with the minimum accumulated benefit (and break ties by choosing the machine with the smallest index). If the job can be scheduled on the machine, do it; if there is no enough residual capacity on that machine, try the machine with the second lowest accumulated benefit (and break ties again by choosing the machine with the smallest index). Skip this job and never consider it again if it cannot be scheduled on any machine.

To help you understand HBF, let's apply it to the instance in Part (b):

i. The result of job sorting is a list of jobs 8, 2, 10, 6, 4, 7, 1, 9, 3, and 5.

ii. Let P_i and B_i be the accumulated processing time and benefit of machine i, the job assignment process is presented in Table 1. Note that the algorithm first tries to schedule job 7 onto machine 1 (because at that moment $B_1 = 9$ is the lowest), but as machine 1 does not have enough risidual capacity (at that moment is $15 - 12 = 3 < p_7 = 5$), job 7 is scheduled onto machine 3 (which has the second lowest accumulated benefit 14 and enough residual capacity $15 - 10 = 5$). Moreover, note that jobs 9 and 3 are skipped because no machine has enough residual capacity for them.

Job	Machine	P	B
8	1	$(12, 0, 0)$	$(9, 0, 0)$
2	2	$(12, 6, 0)$	$(9, 8, 0)$
10	3	$(12, 6, 6)$	$(9, 8, 8)$
6	2	$(12, 14, 6)$	$(9, 15, 8)$
4	3	$(12, 14, 10)$	$(9, 15, 14)$
7	3	$(12, 14, 15)$	$(9, 15, 20)$
1	1	$(15, 14, 15)$	$(14, 15, 20)$
9	–	$(15, 14, 15)$	$(14, 15, 20)$
3	–	$(15, 14, 15)$	$(14, 15, 20)$
5	2	$(15, 15, 15)$	$(14, 18, 20)$

Table 1: The processing of applying HBF to the example instance

iii. The schedule is to assign jobs 1 and 8 to machine 1, jobs 2, 5, and 6 to machine 2, and jobs 4, 7, and 10 to machine 3. The least benefit among all machines is 14 (earned by machine 1).

Write a computer program in any language you like to implement HBF and solve the following instance:

$$m = 4, n = 25, K = 15,$$
$$b = (5, 8, 12, 4, 6, 6, 3, 7, 6, 15, 9, 5, 8, 10, 1, 5, 3, 7, 12, 14, 5, 8, 9, 8, 10), \text{ and}$$
$$p = (3, 6, 7, 5, 4, 2, 6, 3, 5, 8, 10, 2, 4, 7, 1, 5, 8, 3, 6, 4, 12, 4, 8, 4, 7).$$

Write down the solution in a way that a practitioner may understand. List the benefit earned by each machine as well as the least benefit among all machines. Write down the percentage optimality gap with respect to the bound obtained by linear relaxation. Finally, copy and paste your computer program onto your report.

(d) Consider another heuristic algorithm HRF (highest ratio first), which is different from HBF only in the job sorting stage. Now instead of sorting jobs using benefits, please sort jobs using the benefit-processing time ratios (and still break ties according to job indices in the same way). Write a computer program in any language you like to implement HRF and solve the 25-job instance in Part (c). Write down the solution in a way that a practitioner may

understand. List the benefit earned by each machine as well as the least benefit among all machines. Write down the percentage optimality gap with respect to the bound obtained by linear relaxation. Finally, copy and paste your computer program onto your report.

4. (20 points; 10 points each) Consider again Problem 3 and the twenty instances provided in the file OR110-2_hw02_data.zip. In the ZIP file, there are twenty TXT files named in01.txt, in02.txt, ..., and in20.txt. In each TXT file, the first line contains three integers m, n, and K. The second line contains n integers b_1, b_2, ..., and b_n. The third line contains n integers p_1, p_2, ..., and p_n. Two consecutive integers in a line are separated by a white space.[3]

   (a) Write a computer program in any language you like to read these instances, invoke your programs developed in Problems 3c and 3d to solve each instance with the two heuristic algorithms, invoke your program developed in Problem 3b to solve the linear relaxation of each instance, and calculate the percentage optimality with respect to linear relaxation for each instance for each heuristic algorithm. Use a table to list all the fourty optimality gaps. Finally, calculate and compare the average percentage optimality gaps of the two heuristic algorithms. Comment of the performance of the two heuristic algorithms according to the experiment.

   (b) Invent a heuristic algorithm by yourself (e.g., by modifying HBF and HRF). Write down the description of your own algorithm (if you prefer, you may write down the pseudocode). Then convince us that your algorithm is good by doing two things: (1) Intuitively explain why your design makes sense, and (2) redo the experiment in Part (a) to demonstrate the effectiveness of your algorithm.

5. (20 points; 10 points each) Consider the following nonlinear program

$$\min\ 3x_1^2 + 2x_2^2 + 4x_1x_2 + 6e^{x_1} + x_2.$$

   Later when needed, use numerical solutions rather than analytical solutions. For example, when solving $-x = e^x$, using any calculator or software to find $x = -0.567$ as a numerical solution is good enough. There is no need to analytically solve $-x = e^x$.

   (a) Start from $(x_1, x_2) = (0, 0)$ to run one iteration of the gradient descent method to solve this instance. In this iteration, move to the global minimum along the direction you choose. Write down the detailed process of the iteration.

   (b) Start from $(x_1, x_2) = (0, 0)$ to run one iteration of the Newton's method to solve this instance. Write down the detailed process of the iteration.

[3] All these instances are generated such that $\Pr(m = k) = \frac{1}{3}$ for $k \in \{3, 4, 5\}$, $\Pr(n = k) = \frac{1}{5}$ for $k \in \{25, 30, 35, 40, 45\}$, $\Pr(b_j = k) = \frac{1}{20}$ for $k \in \{1, 2, ..., 20\}$, $\Pr(p_j = k) = \frac{1}{20}$ for $k \in \{1, 2, ..., 20\}$, and all numbers are generated independently. K will be chosen as $\left\lceil \frac{3\sum_{j=1}^{n} p_j}{4m} \right\rceil$.