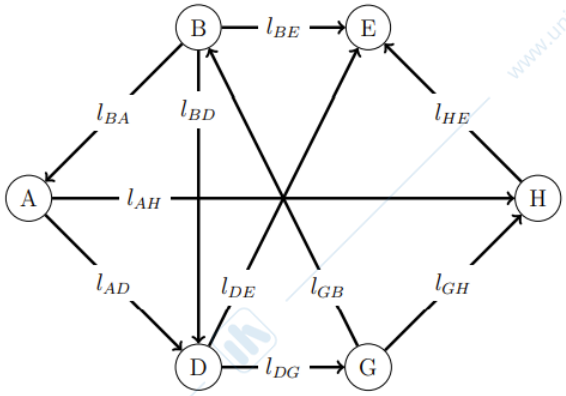
DMO PAST EXAMS' QUESTIONS – LP Exercises

2/2/2018 – Shortest Path

1. Given a directed graph, the Shortest Path Problem (SPP) looks for the minimum-length directed path linking two nodes. A *directed path* from node s to node t is a sequence of directed arcs connecting a sequence of nodes such that:

   - it starts from node s and it ends at node t;
   - if one arc of the path enters into a node (except for s and t), then another arc must leave the same node;
   - a node can appear in the path at most one time.

   Write the LP formulation of the SPP between nodes D and H for the graph in the figure below (in which l_{ij} is the length of arc (i, j)). Moreover, model the following additional requirements on the path: a) if arc (A, D) is chosen in the path, then also arc (G, B) must be; b) at least 3 arcs must belong to the path.



1. **SPP formulation (from node D to node H):**

   Data:

   - l_{ij} : length of arc (i, j)

   Variables:

   - x_{ij} : takes value 1 if arc (i, j) is included in the path, 0 otherwise.

   Model:

$$\min \sum_{(i,j)} l_{ij} x_{ij}$$

   s.t.

$$\sum_{(i,j):i=k} x_{ij} - \sum_{(i,j):j=k} x_{ij} = \begin{cases} 0, & k = A, B, E, G \\ 1, & k = D \\ -1, & k = H \end{cases}$$

$$x_{AD} \leq x_{GB}$$

$$\sum_{(i,j)} x_{ij} \geq 3$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j)$$

28/2/2018 - BTSP

Given a complete directed graph $G := (V, A)$, in which V is the set of nodes and A is the set of arcs, and given a positive cost c_{ij} associated to each arc $(i, j) \in A$, the well-known *Traveling Salesman Problem* (TSP) looks for an Hamiltonian tour in G (i.e., a simple tour that visit all nodes exactly once) having the minimum cost. The so-called *Bottleneck Traveling Salesman Problem* (B-TSP), instead, is a TSP variant in which the objective is to find an Hamiltonian tour in G such that the largest cost of an arc chosen in that tour is as small as possible.
Write a Linear Programming formulation for the B-TSP.

B-TSP formulation:

Data:

- c_{ij} : cost of arc (i, j)

Variables:

- x_{ij} : takes value 1 if arc (i, j) is included in the tour, 0 otherwise.

Model:

$$\min \max_{(i,j)} c_{ij} x_{ij} = \min y$$

s.t.

$$y \geq c_{ij} x_{ij} \quad \forall (i,j) \in A$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V$$

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V$$

$$\sum_{(i,j) \in A : i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, S \neq \emptyset \quad \text{(or any other valid SECs...)}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i,j)$$

$$y \geq 0$$

18/7/2018 – MCKP

Let us consider a set I of items and a knapsack with capacity W. Let p_i and w_i be the profit and the weight, respectively, associated with each item $i \in I$. The well-known Knapsack Problem (KP) is about deciding which of the items should be selected (and loaded into the knapsack) so to maximize the overall collected profit and not to exceed the knapsack capacity. Now consider the partition of set I into k classes, and define I_j as the subset of items belonging to class j, with $j = 1, 2, \ldots, k$. The Multiple-Choice Knapsack Problem (MCKP) is a KP variant in which exactly one item must be selected from each class. Propose a Linear Programming formulation for the MCKP.

MCKP formulation:

Sets and data:

- I: set of items
- k: number of classes for the items
- I_j: subset of items belonging to class j, with $j = 1, 2, \ldots, k$
- w_i: weight of item $i \in I$
- p_i: profit of item $i \in I$
- W: capacity of the knapsack

Variables:

- x_{ij}: takes value 1 if item i of class j is selected, 0 otherwise.

Model:

$$\max \sum_{j=1,2,\ldots,k} \sum_{i \in I_j} p_i x_{ij}$$

s.t.

$$\sum_{j=1,2,\ldots,k} \sum_{i \in I_j} w_i x_{ij} \leq W$$

$$\sum_{i \in I_j} x_{ij} = 1 \quad \forall j = 1, 2, \ldots, k$$

$$x_{ij} \in \{0, 1\} \quad \forall j = 1, 2, \ldots, k, \ \forall i \in I_j$$

19/9/2018 - MKP

Let us consider a set I of items and a set K of knapsacks. Each knapsack $k \in K$ has a different capacity W_k. Moreover, let p_i and w_i be the profit and the weight, respectively, associated with each item $i \in I$. The *Multiple Knapsack Problem* (MKP) is about deciding which of the item to put in which knapsack so to maximize the overall collected profit and not to exceed the knapsacks capacity. Propose an Integer Linear Programming formulation for the MKP.

Sets and data:

- I: set of items
- K: set of knapsacks
- W_k: capacity of the knapsack $k \in K$
- w_i: weight of item $i \in I$
- p_i: profit of item $i \in I$

Variables:

- x_{ik}: takes value 1 if item i is selected and put in knapsack $k \in K$, 0 otherwise.

Model:

$$\max \sum_{k \in K} \sum_{i \in I} p_i x_{ik}$$

s.t.

$$\sum_{i \in I} w_i x_{ik} \leq W_k \quad \forall k \in K$$

$$\sum_{k \in K} x_{ik} \leq 1 \quad \forall i \in I$$

$$x_{ik} \in \{0, 1\} \quad \forall k \in K, \ \forall i \in I$$

6/2/2019 – Minimum Spanning Tree

Consider a connected and undirected graph $G = (V, E)$, where V is the set of nodes and E the set of edges. A *tree* is a connected undirected graph without cycles, a *spanning tree* is a tree that includes all nodes in V.

By considering a cost c_e associated with each edge $e \in E$, the so-called Minimum Spanning Tree (MST) problem looks for a spanning tree of G having the minimum total cost. Provide an (Integer) Linear Programming formulation for the MST problem.

Problem formulation:

Sets and data:

- V: set of nodes
- E: set of edges
- c_e: cost of edge $e \in E$

Variables:

- x_e : takes value 1 if edge $e = [i, j]$ is chosen in the solution tree, and 0 otherwise

Model (SEC-based):

$$\min \sum_{e \in E} c_e x_e$$

s.t.

$$\sum_{e \in E} x_e = |V| - 1$$

$$\sum_{e=[i,j] \in E : i,j \in S} x_e \leq |S| - 1 \quad \forall S \subset V, S \neq \emptyset$$

$$x_e \in \{0, 1\} \quad \forall e \in E$$

26/2/2019 – Covering Supermarkets

Let us consider a set V of towns of a region and a set N of potential locations where to build new supermarkets. A certain number of inhabitants d_i is associated to each town $i \in V$. Inhabitants of a town are said to be *covered* if they do not need to travel more than 15 kms to reach the nearest supermarket. We call C_i the *set of potential covering locations* of town $i \in V$, i.e. the set of all potential locations that are nearer than 15 kms from i. Provide an Integer Linear Programming model to decide where to open exactly p new supermarkets (with $p < |N|$) in order to maximize the number of inhabitants covered.

Problem formulation:

Sets and data:

- V: set of towns
- N: set of potential locations
- C_i: set of potential locations $j \in N$ that cover town $i \in V$
- d_i: inhabitants of town $i \in V$
- p: number of supermarkets to open

Variables:

- y_j is a binary variable taking value 1 if a supermarket is opened in location $j \in N$, and 0 otherwise;
- u_i is a binary variable taking value 1 if inhabitants of town $i \in V$ are covered, and 0 otherwise.

Model:

$$\max \sum_{i \in V} d_i u_i$$

subject to

$$\sum_{j \in N} y_j = p$$

$$u_i \leq \sum_{j \in C_i} y_j \quad \forall i \in V$$

$$y_j \in \{0,1\} \quad \forall j \in N$$

$$u_i \in \{0,1\} \quad \forall i \in V$$

1/7/2019 – PKP

Let us consider an ordered list $I = \{1, 2, \ldots, n\}$ of items and a knapsack with capacity W. Let p_i and w_i be the profit and the weight associated with each item $i \in I$, respectively. The well-known Knapsack Problem (KP) is about deciding which of the items should be selected (and loaded into the knapsack) so to maximize the overall collected profit and not to exceed the knapsack capacity. Now, let us assume that also a penalty π_i is associated to each item $i \in I$. The Penalized Knapsack Problem (PKP) is a KP variant that aims at maximizing the total profit minus the **greatest penalty value** of the selected items, without exceeding the knapsack capacity. Moreover, it allows to select a certain item $i \geq 2$ only if at least another item $j < i$ has been selected. Propose a Linear Programming formulation for the PKP.

PKP formulation:

Sets and data:

- I: set of items
- w_i: weight of item $i \in I$
- p_i: profit of item $i \in I$
- π_i: penalty of item $i \in I$
- W: capacity of the knapsack

Variables:

- x_i: takes value 1 if item i is selected, 0 otherwise
- z: greatest value of penalty of the selected items

Model:

$$\max \sum_{i \in I} p_i x_i - z$$

s.t.

$$\sum_{i \in I} w_i x_i \leq W$$

$$z \geq \pi_i x_i \quad \forall i \in I$$

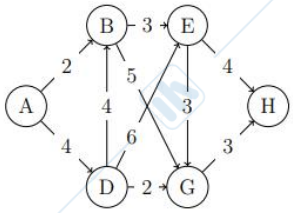
$$x_i \leq \sum_{j \in I, j < i} x_j \quad i \in I, i \geq 2$$

$$x_i \in \{0,1\} \quad \forall i \in I$$

$$z \in \mathbb{R}$$

11/9/2019 – Max Flow

Write the LP formulation of a Max-Flow problem, from the source node A to the sink node H, in the following network (where capacities are written near each arc)



Moreover, ensure that:

- at most one arc between (E, G) and (D, E) has a positive flow;
- if arc (B, G) has a flow strictly greater than 3, then the sum of the flow through arcs (A, D) and (A, B) cannot exceed 4.

Max-Flow formulation:

Sets and data:

- $\mathcal{V}$: set of nodes
- $\mathcal{A}$: set of arcs
- u_{ij}: capacity on arc $(i, j) \in \mathcal{A}$

Variables:

- v := free variable representing the flow passing through the network, from A to H
- x_{ij} := variable representing the amount of flow passing on arc $(i, j) \in \mathcal{A}$
- y_{EG}, y_{DE}:= binary variable taking value 1 if arc (E, G) or (D, E), respectively, has a positive flow, 0 otherwise
- w:= auxiliary binary variable taking value 1 if arc (B, G) has a flow strictly greater than 3, 0 otherwise

Model:

$$\max v$$

s.t.

$$\sum_{j \in \mathcal{V}:(A,j) \in \mathcal{A}} x_{Aj} = v$$

$$\sum_{j \in \mathcal{V}:(i,j) \in \mathcal{A}} x_{ij} - \sum_{j \in \mathcal{V}:(i,j) \in \mathcal{A}} x_{ji} = 0 \quad \forall i \in \mathcal{V} \setminus \{A, H\}$$

$$\sum_{i \in \mathcal{V}:(i,H) \in \mathcal{A}} x_{iH} = v$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in \mathcal{A}$$

$$x_{EG} \leq u_{EG}\, y_{EG}$$

$$x_{DE} \leq u_{DE}\, y_{DE}$$

$$x_{BG} \leq 2w + 3$$

$$x_{AD} + x_{AB} \leq 6 - 2w$$

$$y_{EG} + y_{DE} \leq 1$$

$$x_{ij} \geq 0, (i, j) \in \mathcal{A}$$

$$v \geq 0$$

$$y_{EG}, y_{DE}, w \in \{0, 1\}$$

6/2/2020 – TV ads scheduling

A TV network wants to schedule advertising blocks during the primetime show (21:00-24:00). The primetime is divided into 6 slots, each 30 minutes long. As shown in the Table, each slot has a specific estimated number of viewers. Adverti-

Slot:	1	2	3	4	5	6
Starting time:	21:00	21:30	22:00	22:30	23:00	23:30
Estimated viewers (millions):	3.5	3.9	4.5	4.0	3.5	2.5

sing blocks are of two types: *cut* (3 minutes long) or *extended* (4.5 minutes long). Propose a LP formulation to schedule advertising blocks during the available slots, maximizing the number of viewers and considering that:

- in each slot, only one advertising block can be scheduled;
- the total advertising time for the primetime show cannot exceed 15 minutes.

First note that the solution of the problem is trivial to find. In fact, since the different types of blocks have the same viewers and the main constraint limits the number of available minutes, in order to maximize the number of the scheduled blocks, only *cut*-type blocks will be selected in the optimal solution. In particular, one *cut*-type block will be scheduled in each timeslot.

However, it is still important to have a model in place, which allows to better understand and formalize the problem, and to analyze the possible adding/removing/modification of constraints and objectives.

Notation:
S : set of time slots
v_s : estimated viewers in time slot $s \in S$

Variables:
c_s : equal to 1 if a block of type *cut* is scheduled in slot $s \in S$
e_s : equal to 1 if a block of type *extended* is scheduled in slot $s \in S$

Model:

$$\max \quad \sum_{s \in S} v_s(c_s + e_s)$$

$$c_s + e_s \leq 1, \quad \forall s \in S$$

$$3\sum_{s \in S} c_s + 4.5 \sum_{s \in S} e_s \leq 15$$

$$c_s, e_s \in \{0, 1\}, \quad \forall s \in S$$

20/2/2020 – Incompatibility graph

Let $G = (V, E)$ be an *incompatibility* graph with node set V and edge set $E = E_1 \cup E_2$, where E_1 is a set of non-removable edges and E_2 is a set of removable edges. The presence of an edge $(i, j) \in E$ represents the incompatibility between nodes i and j. However, if $(i, j) \in E_2$, the edge can be removed from the graph (and therefore the relative incompatibility can be eliminated) by paying a penalty cost $c_{ij} > 0$. Considering that a revenue $r_i > 0$ is associated with each node $i \in V$, provide a LP formulation able to find a set of non-incompatible nodes that maximizes the difference between the total revenue associated with the nodes in the chosen set and the total cost associated with the removal of edges between two nodes both chosen in the set.

Variables:
x_i : takes value 1 if vertex $i \in V$ is selected, 0 otherwise;
y_{ij} : takes value 1 if edge $(i,j) \in E_2$ is removed and the associated cost incurred, 0 otherwise.

Model:

$$\max \sum_{i \in V} r_i x_i - \sum_{(i,j) \in E_2} c_{ij} y_{ij}$$
$$x_i + x_j \leq 1, \quad \forall (i,j) \in E_1$$
$$x_i + x_j \leq 1 + y_{ij}, \quad \forall (i,j) \in E_2$$
$$x_i \in \{0,1\}, \quad \forall i \in V$$
$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in E_2$$

23/6/2020 – TSP+

Given a complete directed graph $G := (V, A)$, in which $V = \{1, 2, \ldots, n\}$ is the set of nodes and $A = \{(i,j) : i, j \in V, i \neq j\}$ is the set of arcs, and given a positive cost c_{ij} associated to each arc $(i,j) \in A$, the well-known *Traveling Salesman Problem* (TSP) looks for an Hamiltonian tour in G (i.e., a simple tour that visit all nodes exactly once) having the minimum cost.

Propose a Linear Programming formulation for the TSP also considering the following requirement:

- at least p (where p is a given positive integer number) of arcs having both terminations in the set of nodes $T = \{3, 5, 9\}$ are selected.

What can we say about the problem when $p \geq 3$?

Variables:

- x_{ij} : takes value 1 if arc (i,j) is included in the tour, 0 otherwise.

Model:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V$$
$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V$$
$$\sum_{(i,j) \in A : i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, S \neq \emptyset \text{ (or any other valid SECs...)}$$
$$\sum_{(i,j) \in A : i,j \in T = \{3,5,9\}} x_{ij} \geq p$$
$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A$$

In the case $p \geq 3$, the model will become infeasible.

4/9/2020 – Feasible Flow

Let consider a TLC network in which different information flows, transported at different wavelengths, need to be routed all together. The network can be represented as a directed graph $G = (V, A)$ where V is the set of nodes, and A is the set of arcs representing physical channels between nodes. The set of available wavelengths is W. Each information flow using wavelength $w \in W$ has a specific origin-destination pair of nodes (s^w, t^w), which belong to V, and a specific amount of information d^w to be sent. A cost c_{ij}^w is paid for moving a unit of flow on arc $(i, j) \in A$ at wavelength $w \in W$.

Propose a Linear Programming model able to find a **feasible flow** (i.e., flow conservation must hold) throughout the network, ensuring that:

- each information demand is delivered;
- the bandwidth u_{ij} of each channel $(i, j) \in A$, that must be shared among all the different information flows, is not exceeded;
- the total flow cost is minimized.

Variables:

- x_{ij}^w flow of commodity w on arc (i, j)

Model:

$$\min \quad \sum_{w \in W} \sum_{(i,j) \in A} c_{ij}^w x_{ij}^w$$

$$\text{s.t.} \quad \sum_{j \in V:(i,j) \in A} x_{ij}^w - \sum_{j \in V:(j,i) \in A} x_{ji}^w = +d^w \qquad \forall w \in W, i = s^w$$

$$\sum_{j \in V:(i,j) \in A} x_{ij}^w - \sum_{j \in V:(j,i) \in A} x_{ji}^w = 0 \qquad \forall w \in W, i \in V, i \neq \{s^w, t^w\}$$

$$\sum_{j \in V:(i,j) \in A} x_{ij}^w - \sum_{j \in V:(j,i) \in A} x_{ji}^w = -d^w \qquad \forall w \in W, i = t^w$$

$$\sum_{w \in W} x_{ij}^w \leq u_{ij} \qquad \forall (i, j) \in A$$

$$x_{ij}^w \geq 0 \qquad \forall w \in W, \forall (i, j) \in A$$

DMO PAST EXAMS' QUESTIONS – Theory

04/09/20

3) Discuss the conceptual and technical differences between Two-Stage and MultiStage Stochastic Programming models.

Firstly, the Two-Step SP procedure relies on a "INVEST AND USE" approach. In particular, we could find the following steps:

- First stage: make a set of decisions;
- Recoursive, second stage: the result from the outcome or a "clean-up" process.

On the other hand, for what it may concern the Multi-Stage SP, this can be summarized by an "OPERATIONAL" approach. Moreover, we have the information discovered step by step at a certain time t and the process observed is the following:

- Making a decision at time **t** based on the known informations;
- Studying what is the outcome using the stochastic process in **t;**
- Go on with a t-counter, setting **t=t+1;**
- Repeat from the first step.

Moreover, many types of multistage problems must be re-solved at each stage: solution procedures are recursive versions of two-stage approaches.

However, if the Multi-Stage SP is much reliable to add flexibility in certain models, this is not true with large numbers of stages and branches, making the model difficult to optimize.

Also, all the stages in this procedure are of the same type and a lot of applications could be done by using this kind of approach.

4) Consider the following problem:

$$\min \sum_{i \in I} c_i x_i$$

$$\sum_{i \in I} a_i x_i \geq b$$

$$x_i \in \mathbb{Z}^+, \ \forall i \in I.$$

Make the Lagrangian relaxation of the given constraint and show that the obtained Lagrangian function is a lower bound to the problem.

The main idea is to relax the problem P(x) by removing the "bad" constraints and putting them into the objectivefunction, assigned with weights (theLagrangian multiplier). Each weight represents a penalty which is added toa solution that does not satisfy the particular constraint.

We assume that optimizing over the set *X* can be done very easily, whereas adding the DIFFICULT constraints $\sum_{i \in I} a_i x_i \geq b$ makes the problem very hard to deal with.

Therefore, we introduce a dual variable forevery constraint of Ax ≥ b. The vector μ≥0 is the vector of dual variables (the Lagrangian multipliers) that has the same dimension

as vector *b*. For a fixed μ ≥0, if we let f(x) = $\min \sum_{i \in I} c_i x_i$ and g(x) = $\sum_{i \in I} a_i x_i$, we'll consider the **relaxed** problem as the new objective function:

$$P(x,\mu) = f(x) - \mu(b - g(x))$$

The use of the "-" after the f(x) is the reason of having a Lower Bound of the function, because we want "something less" as the starting P(x) suggests: it's a *min* problem. Furthermore, once we have the Lagrangian Function P(μ) and we want a Lower Bound, the process to maximize and achieve a better LB is to solve a min max problem, in particular:

$$\min \max P(x, \mu)$$

Better known as the *Lagrangian Dual Subproblem*.

23/06/2020

3) Describe the Progressive Hedging algorithm, highlighting the reason why it is very tailored for problems modeled under the Stochastic Programming paradigm.

Progressive Hedging is one of the SP solution methods, that relies on scenario-policy aggregation and Lagrangian Relaxation.

Firstly, the problem is reormulated by expliciting by non-anticipativity constraints the fact that each scenario shares first-stage options with the other scenarios: it relaxes the non-anticipativity constraints through Lagrangian Relaxation. This generates a number of similar subproblems equal to the number of scenarios (then we have to solve them by optimality).

Finally, the **global consesus** represents the stopping condition: it is evaluated and iteratively obtained by updating step by step the Lagrangian Multipliers, possible since we can operate with subgradient method

This procedure is the reason why this all method is called "Hedging": it progressively hedges against the stochasticity of the problem, offering more reliable solutions.

4) Consider the following problem:

$$\min \sum_{i \in I} c_i x_i$$

$$\sum_{i \in I} a_i x_i \geq b$$

$$x_i \in \mathbb{Z}^+, \ \forall i \in I.$$

Make the Lagrangian relaxation of the given constraint and show that the obtained Lagrangian function is a lower bound to the problem.

----- See quest. 4 of 04/09/2020 -----

20/02/2020

3) Explain the meaning of the indicator called Expected Value of the Perfect Information (EVPI) for a Stochastic Linear Programming model (in the minimization form). How is it calculated?

The EVPI represents the LOSS of profit, because of there is a certain stochasticity in the problem: it's the value of how we know the future with certainty.

Also, EVPI is useful in deciding whether to accept additional efforts or not.

The calculation method is the following:

$$EVPI = RP - WS$$

Where *RP* stands for Recourse Problem solution and *WS* for Wait-and-See Solution.

4) Why all NP-complete problems have the same complexity?

In order to answer to this question, we have to introduce the Cook Theorem.

Let be π a problem that belongs to NP and let be SAT a Satisfiability problem:

1. SAT cannot be less complex of any π in NP
2. We can rewrite π as a SAT

From here we are dealing with Logic branch instead of Decision Theory.

Now, a certain π belongs to NP-Complete if:

- π belongs to NP
- there is at least one π' that belongs to NP-Complete, so that π' is reduced to π.

But, since SAT is the "most difficult" problem in NP and since reduction is transitive, we could state that:

SAT is reduced to π' / π' is reduced to π -> SAT is reduced to π

This is the proof in order to demonstrate that π and π' have the same complexity because of the transitivity of reducibility of NP-Complete Problems.

06/02/2020

3) Give an outline of the Simplex Method (do not give the algorithm steps, but the basic idea of the method and its main features).

The general idea of the Simplex Method is the following: given a K feasible set, all the K-vertices represent any possible feasible solution.

Roughly, the steps are:

1. select any basic solution, i.e. any vertex;
2. do a checking phase if our x_bar of interest is an optimal solution
    a. YES = we can stop now, we found the optimum
    b. NO = we simply move to another one (only if is a basic one)
3. select a new basic feasible solution

In order to get a new base the key idea is to swapping two vectors of a certain matrix A, which is divided, for example, into submatrix B and D: these two have the vector that we want to swap between them and more precisely:

B: index from 1 to m; D: index from m+1 to n

This updating procedure must rely on a precise criterion: the **ε** factor that multiplies the equation in order to test the swapping must be positive and in particular

$$\varepsilon = \min \frac{x_i}{y_{iq}} \qquad \text{where } 1 <= i <= m$$

Also, the two terms of the fraction have to be positive, otherwise, if all y_{iq} are negative, we have no feasible solution (problem unbounded).

Now, the situation is:

- we have a vector that is leaving the base
- we have to check **ε** (if it's wrong, solution is infeasible)
- a new vector is entering the new base

A good example of the simplex method is the Diet Problem. Here we could observe that a type of food is "entering or not" in our trolley only if its nutrients are acceptable (in terms of cost and value) with respect to what we have already. This choice is good if I spend less than leaving the product of interest on the shelf, otherwise, what I've in the trolley is good enough and so this certain product can

remain where it is. To conclude the example, another situation is considered: if after a certain time I continuously decide not to bring other food from the shelf it means that what I have is already the optimal solution.

4) Write the first order necessary conditions for a local minimum of a non-linear minimization problem subject to equality constraints.

11/09/2019

3) Define the complexity classes P, NP and NP –Complete. What is the relationship between these classes?

P is set of problems that can be solved in Polynomial time.

NP is set of decision problems that can be solved in Non Polynomial time. P is subset of NP any problem that can be solved by deterministic way in polynomial time can also be solved by non-deterministic way in polynomial time)

NP-complete problems are the hardest problems in NP set. A decision problem π is NP-complete if:
1) π is in NP (Any given solution for NP-complete problems can be verified quickly, but there is no efficient known solution).
2) Every problem in NP is reducible to π in polynomial time.

A problem is NP-Hard if it follows property 2 mentioned above, doesn't need to follow property 1. Therefore, NP-Complete set is also a subset of NP-Hard set.

01/07/2020

3) Define the concept of problem reducibility and show how the Partition problem can be reduced to the Knapsack problem.

Reducibility is a very useful concept, whose purpose is to let a generic problem π' be a *special case* of π because the latter is at most complex than the π' (so π' is at least complex than π). Also, this property can be used to achieve the statement of "same complexity" between two open problems π and π', both belonging to the class NP-complete.

An important application in order to fully understand this criterion relies on the Subset Sum Problem (Partition Problem) and Knapsack Problem: in both problem we have to get a sum, with the respective constraints.

$$max \sum x_i \cdot p_i$$

$$\sum w_i \cdot x_i \leq W$$

Nevertheless, what we wanto to know is also if we are dealing with the same problem procedure: to answer this, we could imagine that the constraint W is like a segment $\boldsymbol{b}$ and consider either the profit p and the weight w like sub-segments $\boldsymbol{a_i}$.

In this general way the problem is rewritten as:

$$z^* = max \sum x_i \cdot a_i$$

$$\sum a_i \cdot x_i \leq b$$

So:

- if z^*< b, also Partition problem is solved because there is no any subset sum available;
- if z^* = b, we still have solved Partition Problem.

Finally, we could state that any istance of Partition Problem derived in polynomial time, there would be the respective istance of Knapsack Probem so that solving KP we solve Partition Problem: i.e. the latter is reducible to a KP.

26/02/2019

3) Describe the optimality principle which the dynamic programming is based on. Give an example.

Dynamic Programming belongs to the family of Exact Omptimization Method and has two main features:

1. decomposition of problem into a family of similar but easier problems;

2. recursive formulation.

The principle on which this method relies on is the Bellman Principle: given an optimal solution with its Shortest Path (from a source **s** to node **i**) and a subpath from **s** to **p** (generic node before i), it states that this subpath is the *Shortest Path* for the subproblem s->p. Generally speaking, we can say that *pieces of optimal solutions* are themselves optimal.

A possible application is the KP 0/1, which model is a simple maximum problem with its respective constraints, but here every item could be 0 or 1. The key idea is to think about this problem as a bunch of j-subproblems: we have to decide to load or not the **j-item**, having in this way n-subproblems; most important, we don't know for an item **i** different from j what's happening: if we do, we make an heuristic and it's not good.

06/02/2019

3) What is the philosophy of the Simplex method? What is the complexity of the Simplex method in the worst case? Why?

---------- See quest.3 of 6/2/20----------

19/09/2018

3) Describe how the Simplex method does work. Moreover, precise which kind of problems it is able to solve, and elaborate on its computational complexity

---------- See quest.3 of 6/2/20----------

+

The Simplex Method let the user go from a basic solution of a certain min/max problem to another baisc solution in order to reduce/increase iteratively the objective function value until its minimum/maximum would be achieved.

18/07/2018

3) Consider the following problem:

$\min \sum_{i \in I} c_i x_i$

$\sum_{i \in I} a_i x_i \geq b$

$x_i \in \mathbb{Z}^+, \ \forall i \in I.$

Make the Lagrangian relaxation of the given constraint and show that the obtained Lagrangian function is a lower bound to the problem.

----------See quest.4 of 4/9/20----------

28/02/2018

3) Explain how the simplex method does work, its advantages and disadvantages

---------- See quest.3 of 6/2/20----------

+

Pros of simplex:

- Given *n* decision variables, usually converges in *O*(*n*) operations
- Takes advantage of geometry of problem: visits vertices of feasible set and checks each visited vertex for optimality. (In primal simplex, the reduced cost can be used for this check.)
- Good for small problems.

Cons of simplex:

- Given *n* decision variables, you can always find a problem instance where the algorithm requires *O*(2*n*) operations to arrive at a solution.
- Not so great for large problems, because pivoting operations become expensive.

02/02/2018

3) Let us consider the Dynamic Programming method. What is the optimality principle this method is based on? Write the Dynamic Programming recursive relationship for the Knapsack Problem 0/1.

----------See quest.3 of 26/2/19----------

+

Recalling the Shortest Path Problem, when I select a node j, I have to consider also its predecessors and assuming known the Shortest Path of each of them: that's why we observe a recursive/nested structure.

Now, since an interesting application is the KP 0/1 we can apply this recursive process to it, too. For any existing sub-problem, the maximum profit is given by:

$$F^* = F_1(0)$$

This suggests us to start from the last item in order to arrive to $F_1(0)$. So, for a generic E-step, we have:

1. $F_n(E) = p_n$ if the content is LOADED
2. $F_n(E) = 0$,otherwise

Summarizing, we obtain a general formula that would choose the max between **two arguments** that is the following:

$$F_j(E) = max\left(F_{j+1}(E), p_j + F_{j+1}(E + w_j)\right)$$

DMO Simulation's questions

What is the philosophy of the Ant Colony Optimization method?

Ant Colony Optimization method belongs to Multi Start class of Metaheuristics. The baic idea relies on the method used by the ants to create the shortest path to the food from their nest.

So, the ants produce a trial made up by their pheromone: this will disappear after a while, so this means that in terms of time and space length, pheromones on the shortest path would disappear later.

Finally, in the last moment we find the majority of ants that will follow the more-pheronome path, i.e. the shortest: e.g., we have two sides in which the right one is the shortest. 10 ants go on the right and 10 on the left; after certain time the right-ants will come back quicker than the left ones, laying a trial of pheromones stronger than before. After these ants returned to their nest, the next time more ants will go on right side, i.e. the one that achieved more pheromones.

This philosophy can be applied on a certain graph path, working with "virtual ants" that cross each node, in order to better understand what are their features:

- ant chooses the town to move into with a certain probability

- the subtours are removed (thanks to the creation of a tabu list, so that we store nodes already visited)
- when ant completes a tour, it lays a trial of each edge, like a mark
- we begin with the iterations, moving M ants in the interval (t, t+1). After N iterations each ant made a tour

In what does it consist of the Recovery step in the Recovery Beam Search method?

In Recovery Step the aim is to recover better solution in improving lower bounds starting from a partial solution, by generating a new neighbour of this partial solution through two different steps:

- by swapping a value (for this we have O(n^2) complexity)
- by changing a value: so we generate a neighbour by changing a value from 0 to 1 (complexity equal to O(n)).

The risk of doing so is that generating a new neighbour by swapping or changing a value can bring us to a infeasible solution so we have to check feasibility in order to proceed.

Present and discuss the Karush-Kuhn-Tucker conditions for inequality constraintsin non-linear programming

Karush-Kunn-Tucker conditions are necessary conditions in order to solve a Non Linear Programming problem and this approach relies on a Lagrangian process. It's also applied to a minimization non linear problem in which we have:

min f(x) (O.F.)

g_i(x) ≤ 0

h_j(x) = 0

Let be x* a local minimum and f,g,h functions that can be differtiable in x*, then:

$$1.\ \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{l} \mu_j \nabla h_j(x^*) = 0,$$
$$2.\ g_i(x^*) \leq 0, \forall i = 1, \ldots, m$$
$$3.\ h_j(x^*) = 0, \forall j = 1, \ldots, l$$
$$4.\ \lambda_i \geq 0\ (i = 1, \ldots, m)$$
$$5.\ \lambda_i g_i(x^*) = 0\ \forall i = 1, \ldots, m.$$

Where the first equation is the null condition for the gradient of the lagrangian part of the problem; then for 2. and 3. we have the strict costraints regarding x*; then we have the non negtivity condition related to the coefficient and finally we have the nullify of an “inactive” costraint.

What is an aspiration criterion in Tabu Search and how does it work?

Tabu Search is one of the methods belonging to Neighborhood Search Metaheuristics. In this method we escape from the local optima avoiding the visit of previou solutions for a certain number of iterations and we use steepest descent local search approach.

The reason we use the term “tabu” is referred to a set of prohibited moves that can bring to an already visited solution.

Once we create a tabu list, we rely on the aspiration criterion if we can accept a tabu move only if that generates the best solution founded until the actual iteration: we don’t throw away the optimum.