Algorithmic Methods for Mathematical Models

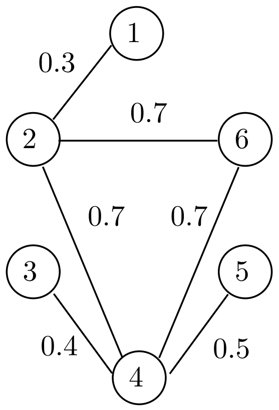
– COURSE PROJECT –

A surveillance company is developing a program for shape recognition. For the sake of simplicity, let us assume that a surveillance system consists of a camera, which can take pictures, and a distance sensor. When the camera captures an image, the goal is to determine if a particular shape (a person, an animal, etc.) appears in it, so that it can be decided whether an alarm should be triggered or not. The distance sensor provides auxiliary information to that end.

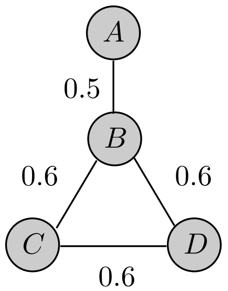
More precisely, an *image* is modelled with an undirected graph $G = (V, E)$, where each vertex $v \in V$ is a point in the image and $E \subseteq V \times V$. An edge $\{u, v\} \in E$ is placed between points u and v according to an edge detection algorithm that has already been implemented by the company. Moreover, edges E are weighted: there is a function $\omega : E \to (0, 1)$ such that $\omega(u, v)$ is the Euclidean distance between points u and v as measured with the distance sensor (actually, between the points in the 3D space whose projections are u and v on the image; but this detail is not rellevant for the purposes of this project). Assume that distances are scaled so that they are always less than 1.

Moreover, a *shape* is also modelled with a weighted undirected graph $H = (W, F)$, where $F \subseteq W \times W$, with weight function $\rho : F \to (0, 1)$. We say that a shape $H = (W, F)$ *occurs* in an image $G = (V, E)$ when there is an injective function $f : W \to V$ (called the *embedding* of H in G) that is edge-preserving: i.e., $\{x, y\}$ is an edge in H if and only if $\{f(x), f(y)\}$ is an edge in G. In order to discard spurious cases, we are particularly interested in the embeddings that minimize the sum of absolute differences between the weight of an edge e and the weight of $f(e)$. The goal of this project is, given an image and a shape, to find an optimal embedding according to this criterion.

For example, let us consider the following image (left) and shape (right):

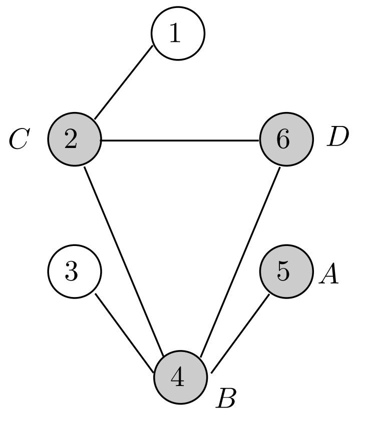


Image

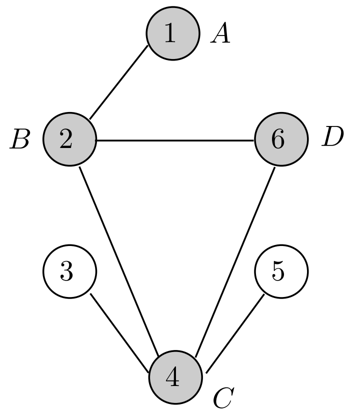


Shape

In this case $V = \{1, 2, 3, 4, 5, 6\}$ and $W = \{A, B, C, D\}$. Possible embeddings are $f : W \to V$ defined as $f(A) = 5$, $f(B) = 4$, $f(C) = 2$ and $f(D) = 6$, or $f' : W \to V$ defined as $f'(A) = 1$, $f'(B) = 2$, $f'(C) = 4$ and $f'(D) = 6$. These embeddings represent the following occurrences of the shape in the image, respectively:



Embedding f



Embedding f'

Moreover f is a better embedding than f', as the sum of absolute differences of weights is $|0.5-0.5|+|0.7-0.6|+|0.7-0.6|+|0.7-0.6| = 0.3$ for the former, and $|0.3-0.5|+|0.7-0.6|+|0.7-0.6|+|0.7-0.6| = 0.5$ for the latter.

On the other hand, the function $f'' : W \to V$ defined as $f''(A) = 1$, $f''(B) = 6$, $f''(C) = 4$ and $f''(D) = 2$ is not an embedding as (among other reasons) $\{A, B\} \in F$ and but $\{f''(A), f''(B)\}) = \{1, 6\} \notin E$.

1. **Work to be done:**

   (a) State the problem formally. Specify the inputs and the outputs, as well as any auxiliary sets of indices that you may need, and the objective function.

   (b) Build an integer linear programming model for the problem and implement it in OPL.

   (c) Because of the complexity of the problem, heuristic algorithms can also be applied. Here we will consider the following:

      i. a greedy constructive algorithm,

      ii. a greedy constructive + a local search procedure,

      iii. GRASP as a meta-heuristic algorithm. You can reuse the local search procedure that you developed in the previous step.

   Design the three algorithms and implement them in the programming language you prefer.

   (d) Tuning of parameters and instance generation:

   Given an instance consisting of an image $G = (V, E)$ and a shape $H = (W, F)$, we define its *size* as $|V|$, and its *load* as $\frac{|W|}{|V|}$.

   Moreover, let us recall that the *density* δ of a graph with N vertices and M edges is $\delta = \frac{M}{\binom{N}{2}}$.

      i. Implement an instance generator that generates random instances for given values of $|V|$, $|E|$, $|W|$ and $|F|$.

      ii. Tune the α parameter of the GRASP constructive phase with a set of feasible randomly generated instances of large enough size.

      iii. Study the time that CPLEX needs to solve a set of mid-sized instances: Generate feasible instances with the same size but different values for load and densities of G and H, and solve them. The minimum time to solve these instances has to be at least 10 min. Select the combination of values for the load λ and the densities $\delta(G)$ and $\delta(H)$ that take the maximum time to solve.

   iv. Generate feasible problem instances with increasingly larger size but with the same load λ and densities $\delta(G)$ and $\delta(H)$ found in the previous step. Solving each instance with CPLEX should take from 1 to 30 min.

(e) Compare the performance of solving the model with CPLEX with applying the heuristic algorithms, both in terms of computation time and of quality of the solutions as a function of the size of the instances.

(f) Prepare a report and a presentation of your work on the project.

2. **Report:**

   Prepare a report (8-10 pages) in PDF format including:

   - The formal problem statement.
   - The integer linear programming model, with a definition and a short description of the variables, the objective function and the constraints. Do not include OPL code in the document, but rather their mathematical formulation.
   - For the meta-heuristics, the pseudo-code of your constructive, local search, and GRASP algorithms, including equations for describing the greedy cost function(s) and the RCL.
   - Tables or graphs with the tuning of parameters and instance generation.
   - Tables or graphs with the comparative results.

   Together with the report, you should also give all sources (OPL code, programs of the meta-heuristics, instance generator, etc.) and instructions on how to use them, so that results can be easily reproduced.

3. **Presentation:**

   You are expected to make a presentation of your work (7-10 minutes long) at the end of the course.

   The slots of Tuesday 14/12/21, Friday 17/12/21 and Tuesday 21/12/21 will be devoted to these presentations. The schedule will be announced in its due time.

   The slides of the presentation in PDF format should be delivered together with the report by **Monday 13/12/21**.

   The idea is that the presentation can contain figures, plots, equations, algorithms, etc. with a very short text that helps to understand them. It is expected that you give a full explanation of those contents during your presentation. On the other hand, the report should contain that explanation in a well-organized manner as a text.