Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Algoritmos Avançados

2025/2026 — 1º Semestre

1st Project — Exhaustive Search

Deadline: November 10, 2025

Introduction

Design and test an **exhaustive search algorithm** to solve one of the following **graph problems**, as well as **another method using a greedy heuristic**.

Afterwards, analyze the computational complexity of the developed algorithms. To accomplish that:

a) Perform a formal computational complexity analysis of the developed algorithms.

b) Carry out a sequence of experiments, for successively larger problem instances, to register and analyze (1) the **number of basic operations** carried out, (2) the **execution time** and (3) the **number of solutions / configurations** tested, as well as (4) the **precision of the greedy heuristic** used.

c) Compare the results of the experimental and the formal analysis.

d) Determine the **largest graph** that you can process on your computer using the **exhaustive search** algorithm, without taking too much time.

e) Estimate the execution time that would be required by much larger problem instances.

f) Write a report (8 pages, max.).

Graphs for the Computational Experiments

The **graph instances** used in the **computational experiments** should represent the following scenario:

-   graph **vertices** are 2D points on the XOY plane, with integer valued **coordinates** between 1 and 500.

-   graph **vertices should neither be coincident nor too close**.

-   the **number of edges sharing a vertex is randomly determined**.

Generate successively larger **random graphs**, with 4, 5, 6, … vertices, using your **student number** as **seed**.

For each **fixed number of vertices**, generate **graph instances with 12.5%, 25%, 50% and 75% of the maximum number of possible edges** for that number of vertices.

Suggestions:
- use Python's **NetworkX** package: https://networkx.org/
- store each graph in a file to be used for the computational experiments.

Depending on the problem, it might be helpful to represent each graph by its **adjacency matrix** or by its **incidence matrix** or by any **other representation** you might consider useful.

It might also be useful to graphically **visualize** the problem instances and the computed solutions.

Decision Problems – Set k to 12.5%, 25%, 50% and 75% of the number of graph vertices

1 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **clique with k vertices**? A **clique of G** is a subset of vertices, all adjacent to each other, i.e., defining a complete subgraph of G. Example: in a social network, a clique is a subset of people who all know each other.

2 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have an **independent set with k vertices**? An **independent set of G** is a subset of vertices, no two of which are adjacent. Example: in a social network, an independent set is a subset of people who do not know each other.

3 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **vertex cover with k vertices**? A **vertex cover of G** is a set C of vertices, such that each edge of G is incident to, at least, one vertex in C.

4 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **dominating set with k vertices**? A **dominating set of G** is a subset D of vertices, such that every vertex not in D is adjacent to, at least, one vertex in D. Application example: dominating sets are used in wireless networking to find efficient routes.

5 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **connected dominating set with k vertices**? A **connected dominating set of G** is **(1)** a subset D of vertices, such that every vertex not in D is adjacent to, at least, one vertex in D; and **(2)** any vertex in D can reach any other vertex in D by a path that stays entirely within D. That is, D induces a connected subgraph of G.

6 – For a given **directed graph** $G(V, E)$, with n vertices and m edges, does G have a **closure with k vertices**? A **closure of G** is a set of vertices C, such that no edges leave C.

Decision Problems – Set k to 12.5%, 25%, 50% and 75% of the number of graph edges

7 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have an **edge cover with k edges**? An **edge cover** is a set C of edges such that each vertex of G is incident to, at least, one edge in C.

8 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have an **edge dominating set with k edges**? An **edge dominating set of G** is a subset D of edges, such that every edge not in D is adjacent to, at least, one edge in D.

9 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **cut with k edges**? A **cut of G with k edges** is a partition of the graph's vertices into two complementary sets S and T, such that the number of edges between the set S and the set T is k.

10 – For a given undirected graph $G(V, E)$, with n vertices and m edges, does G have a **matching with k edges**? A **matching in G** is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex.

Optimization Problems

11 – Find a **maximum clique** for a given undirected graph $G(V, E)$, with n vertices and m edges. A **clique of G** is a subset of vertices, all adjacent to each other, i.e., defining a complete subgraph of G. A **maximum clique** is a clique with the **largest** possible number of vertices. Example: in a social network, a clique is a subset of people who all know each other.

12 – Find a **maximum weight clique** for a given undirected graph $G(V, E)$, whose **vertices carry positive weights**, with n vertices and m edges. A **clique of G** is a subset of vertices, all adjacent to each other, i.e., defining a complete subgraph of G. The **weight of a clique** is the sum of its vertices' weights. A **maximum weight clique** is a clique whose total weight is as large as possible.

13 – Find a **maximum independent set** for a given undirected graph $G(V, E)$, with n vertices and m edges. An **independent set of G** is a subset of vertices, no two of which are adjacent. A **maximum independent set** is an independent set of **largest** possible size. Example: in a social network, an independent set is a subset of people who do not know each other.

14 – Find a **maximum weight independent set** for a given undirected graph $G(V, E)$, whose **vertices carry positive weights**, with n vertices and m edges. An **independent set of G** is a subset of vertices, no two of which are adjacent. The **weight of an independent set** is the sum of its vertices' weights. A **maximum weight independent set** is an independent set whose total weight is as large as possible.

15 – Find a **minimum vertex cover** for a given undirected graph $G(V, E)$, with n vertices and m edges. A **vertex cover of G** is a set C of vertices, such that each edge of G is incident to, at least, one vertex in C. A **minimum vertex cover** is a vertex cover of **smallest** possible size.

16 – Find a **minimum weight vertex cover** for a given undirected graph $G(V, E)$, whose **vertices carry positive weights**, with n vertices and m edges. A **vertex cover of G** is a set C of vertices, such that each edge of G is incident to, at least, one vertex in C. The **weight of a vertex cover** is the sum of its vertices' weights. A **minimum weight vertex cover** is a vertex cover whose total weight is as small as possible.

17 – Find a **minimum dominating set** for a given undirected graph $G(V, E)$, with n vertices and m edges. A **dominating set of G** is a subset D of vertices, such that every vertex not in D is adjacent to, at least, one vertex in D. A **minimum dominating set** is a dominating set of **smallest** possible size. Application example: dominating sets are used in wireless networking to find efficient routes.

18 – Find a **minimum connected dominating set** for a given undirected graph $G(V, E)$, with n vertices and m edges. A **connected dominating set of G** is **(1)** a subset D of vertices, such that every vertex not in D is adjacent to, at least, one vertex in D; and **(2)** any vertex in D can reach any other vertex in D by a path that stays entirely within D. That is, D induces a connected subgraph of G. A **minimum connected dominating set** is a connected dominating set of **smallest** possible size.

19 – Find a **minimum weight dominating set** for a given undirected graph $G(V, E)$, whose **vertices carry positive weights**, with n vertices and m edges. A **dominating set of G** is a subset D of

vertices, such that every vertex not in *D* is adjacent to, at least, one vertex in *D*. The **weight of a dominating set** is the sum of its vertices' weights. A **minimum weight dominating set** is a dominating set whose total weight is as small as possible.

20 – Find a **minimum edge cover** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. An **edge cover** is a set C of edges such that each vertex of G is incident to, at least, one edge in *C*. A **minimum edge cover** is an edge cover of **smallest** possible size.

21 – Find a **minimum weight edge cover** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. An **edge cover** is a set C of edges such that each vertex of G is incident to, at least, one edge in *C*. The **weight of an edge cover** is the sum of its edges' weights. A **minimum weight edge cover** is an edge cover whose total weight is as **small** as possible.

22 – Find a **minimum edge dominating set** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. An **edge dominating set of *G*** is a subset *D* of edges, such that every edge not in *D* is adjacent to, at least, one edge in *D*. A **minimum edge dominating set** is an edge dominating set of **smallest** possible size.

23 – Find a **minimum weight edge dominating set** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. An **edge dominating set of *G*** is a subset *D* of edges, such that every edge not in *D* is adjacent to, at least, one edge in *D*. The **weight of an edge dominating set** is the sum of its edges' weights. A **minimum weight edge dominating set** is an edge dominating set whose total weight is as **small** as possible.

24 – Find a **maximum cut** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **maximum cut of *G*** is a partition of the graph's vertices into two complementary sets S and T, such that the number of edges between the set S and the set T is as **large** as possible.

25 – Find a **maximum weighted cut** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **maximum weighted cut of *G*** is a partition of the graph's vertices into two complementary sets S and T, such that the **sum of the weights of edges between the set S and the set T** is as large as possible.

26 – Find a **minimum cut** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **minimum cut of *G*** is a partition of the graph's vertices into two complementary sets S and T, such that the number of edges between the set S and the set T is as **small** as possible.

27 – Find a **minimum weighted cut** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **minimum weighted cut of *G*** is a partition of the graph's vertices into two complementary sets S and T, such that the **sum of the weights of edges between the set S and the set T** is as small as possible.

28 – Find a **maximum weighted closure** for a given vertex-weighted directed graph *G(V, E)*, with *n* vertices and *m* edges. A **closure of *G*** is a set of vertices *C*, such that no edges leave *C*. The **weight of a closure** is the sum of its vertices' weights. A **maximum weight closure** is a closure whose total weight is as large as possible.

29 – Find a **minimum weighted closure** for a given vertex-weighted directed graph *G(V, E)*, with *n* vertices and *m* edges. A **closure of *G*** is a set of vertices *C*, such that no edges leave *C*. The **weight of a closure** is the sum of its vertices' weights. A **minimum weight closure** is a closure whose total weight is as small as possible.

30 – Find a **maximum matching** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **matching in *G*** is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex. A **maximum matching** is a matching of **largest** possible size. Application example: computational chemistry.

31 – Find a **maximum weighted matching** for a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. A **matching in *G*** is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex. A **maximum weighted matching** is a matching for which the **sum of the weights of its edges** is as large as possible.

32 – Find the **chromatic number** of a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. The **chromatic number of *G*** is the **smallest** number of colors needed to properly **color the vertices** of graph *G*, i.e., to label the vertices with colors such that no two adjacent vertices have the same color. Application example: vertex coloring is used for register allocation in compilers.

33 – Find the **chromatic index** of a given undirected graph *G(V, E)*, with *n* vertices and *m* edges. The **chromatic index of *G*** is the **smallest** number of colors needed to properly **color the edges** of graph G, i.e., to label the edges with colors such that no two edges sharing the same vertex have the same color. Application example: edge coloring is used for frequency assignment, for fiber optics networks.

J. Madeira, October 10, 2025