Department of Telecommunications, electronic and physic

Course: Operational Research, theory and applications

Laboratory Report

Academic Year 2024/2025

Students:

Marco Corso - s344060
Lorenzo Cavallaro - s346742
Lorenzo Eustachio Braia - s346316

June 5, 2025

Contents

1 LAB 1: Graph coloring

1.1 Problem Description

The graph coloring problem consists of assigning a color to each node of an undirected graph so that no two adjacent nodes share the same color. The main objective is to minimize the total number of colors used.

1.2 Problem Implementation

Let $E(V, V)$ be the adjacency matrix of the graph, where V is the vector of nodes. Since the graph is undirected, E is symmetric. Each entry $E(u, v) = 1$ if there is an edge that connects nodes u and v (equally $E(v, u) = 1$), while the value is 0 if there is no connection between the two nodes.
Being C a set of integers, where each number represents a color, the vector $y(C)$ have value 1 in the position c if the color c is used in the graph, otherwise it is 0. The matrix $x(V, C)$ has value 1 if the color c is used for coloring the node v, 0 if not. The goal is to minimize the number of color, given that adjacent nodes must have different colors.
$min(G = \sum_{c=1}^{C} y(c))$ is the objective function that minimize the number of colors used.

1.3 Constraints

- each node must be colored with exactly one color:
  $\sum_{c=1}^{C} x(v, c) = 1 \ \forall v \in V$
- adjacent nodes must have different colors:
  $x(u, c) + x(v, c) \leq 1 \ \ \forall \ u, v \in \ V \mid (u, v) \in E \ \forall c \in C$
- if a color is used for a node, it must be marked as used:
  $x(v, c) \leq y(c) \ \forall v \in V \ \forall c \in C$

1.4 Results

The result is obtained by minimizing the objective function. Since graph coloring is a computationally difficult problem and belongs to the class of NP-hard problems, finding an optimal solution often requires exploring a large number of possible combinations. To reduce the computational effort, we limited the iterations to the upper triangular part of the adjacency matrix and reduced the size of the available color set.
However, the problem remains challenging, especially as the number of nodes and the edge probability increase.
We illustrate our results with a plot based on a graph containing 20 nodes and an edge probability of 0.3.

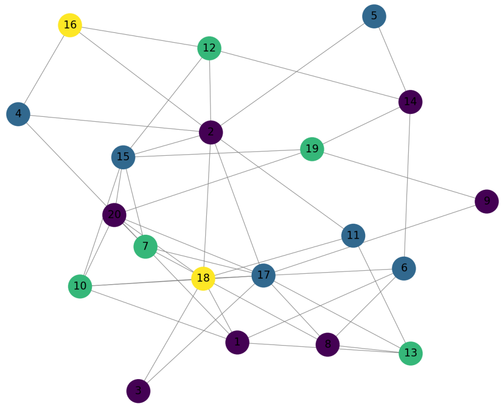

Figure 1: **Graph N=20, p=0.3**

2 LAB 2: Maximum clique

2.1 Problem Description

The maximum clique problem aims to find the largest set of nodes in a graph such that every pair of nodes in the set is connected by an edge. Formally, a clique is a fully connected subgraph, and the goal is to identify the clique with the maximum cardinality.

2.2 Problem Implementation

Let $E(V, V)$ be the adjacency matrix of the graph, where V is the vector of nodes. Since the graph is undirected, E is symmetric. Each entry $E(u, v) = 1$ if there is an edge that connects nodes u and v (equally $E(v, u) = 1$), while the value is 0 if there is no connection between the two nodes.
$x(V)$ is a V-dimensional array where $x(v) = 1$ if the node v is inside the clique.
The goal of the problem is to maximize the number of nodes inside the clique set.
$max(G = \sum_{v=1}^{V} x(v))$ is the objective function that maximize the number of nodes inside the clique.

2.3 Constraints

- nodes connected by an edge can be inside the same clique:
  $x(u) + x(v) \leq 2 \;\; \forall u, v \in V \mid (u, v) \in E$
- nodes that are not connected between them must not be in the clique:
  $x(u) + x(v) \leq 1 \;\; \forall u, v \in V \mid (u, v) \notin E$

2.4 Results

The result is obtained by maximizing the objective function. In this problem we avoid to use a matrix as decision variable because we need only to save if a node is or not inside the clique set. This helps the problem to be lighter in terms of computational cost. There is a correlation between the maximum clique set and the graph coloring, the number of colors used is always greater or equal to the size of

the largest clique set, because the nodes inside the clique need a different color for each.
We illustrate our results with a plot based on a graph with 20 nodes and an edge probability of 0.1.

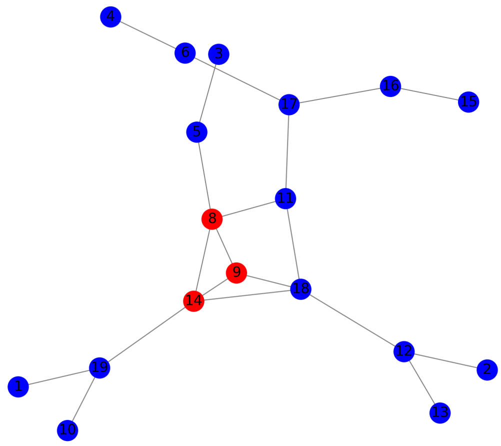

Figure 2: **Graph N=20 p=0.1**

3 LAB 3: Maximum independent set

3.1 Problem Description

The maximum independent set problem aims to find the largest set of nodes in a graph such that every pair of nodes in the set is not connected by an edge. The maximum independent set in a graph G is equivalent to the maximum clique in its complement graph $\bar{G}$ (with flipped edges). This problem is also related to the graph coloring, because the nodes colored with the same color compose an independent set, so minimizing the number of colors is equivalent to cover the graph with the fewest number of independent sets.

3.2 Problem Implementation

Let $E(V, V)$ be the adjacency matrix, where V is the vector of nodes. Since the graph is undirected, E is symmetric. Each entry $E(u, v) = 1$ if there is an edge that connects nodes u and v (equally $E(v, u) = 1$), while the value is 0 if there is no connection between the two nodes.
$x(V)$ is a V-dimensional array where $x(v) = 1$ if the node v is inside the independent set. The goal of the problem is to maximize the number of nodes inside the independent set.
$max(G = \sum_{v=1}^{V} x(v))$ is the objective function that maximize the number of nodes inside the independent set.

3.3 Constraints

- nodes connected by an edge cannot be in the same independent set:
  $x(u) + x(v) \leq 1 \ \ \forall\, u, v \in \ V \mid (x, v) \in E$
- nodes that are not connected can be in the same independent set:

  $x(u) + x(v) \leq 2 \ \ \forall\, u, v \in \ V \mid (x, v) \notin E$

3.4 Results

The result is obtained by maximizing the objective function. In this problem we avoid to use a matrix as decision variable because we need only to save if a node is or not inside the independent set. This helps the problem to be lighter by computational point of view.
We illustrate our results with the plot of a graph with 20 nodes and an edge probability of 0.3.

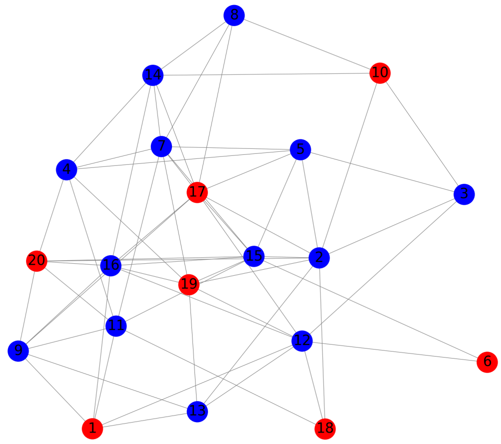

Figure 3: **Graph N=20 p=0.3**

4 LAB 4: Graph coloring revisited

4.1 Problem description

The Graph coloring revisited problems is similar to a regular Graph Coloring, but treats colors as frequencies. The constraints require that adjacent nodes cannot have the same color and cannot have adjacent colors too. Moreover, the goal of the problem is to minimize the bandwidth of the colors used.

4.2 Problem Implementation

Let $E(V, V)$ be the adjacency matrix of the graph, where V is the vector of nodes. Since the graph is undirected, E is symmetric. Each entry $E(u, v) = 1$ if there is an edge that connects nodes u and v (equally $E(v, u) = 1$), while the value is 0 if there is no connection between the two nodes.
Being C the set of colors available, we create the boolean matrix $x(V, C)$ that stores 1 where the node v is colored with c. Then we have also the boolean vector $y(C)$ that contains 1 in the position c if the color c is used in the graph, otherwise it is 0.
To evaluate the bandwidth we should know the minimum and the maximum color used. In this solution we assume that the minimum bandwidth is always 1, then we can compute the maximum color as the minimum value higher than all the possible products $c \cdot y(c)$.
The goal is to minimize the bandwidth of the colors used, given that adjacent nodes must have colors that are different and not adjacent in the color set.
$min(G = maxColor - 1))$ is the objective function that minimizes the bandwidth of the colors used.

4.3 Constraints

- each node must be colored with exactly one color:
  $\sum_{c=1}^{C} x(v, c) = 1 \ \forall v \in V$
- adjacent nodes must have different colors:
  $x(u, c) + x(v, c) \leq 1, \quad x(u, c) + x(v, c-1) \leq 1, \quad x(u, c) + x(v, c+1) \leq 1$

$\forall\ u, v \in\ V \mid (u, v) \in E\ \forall c \in C$

- if a color is used for a node, it must be marked as used:
  $x(v, c) \leq y(c)\ \forall v \in V\ \forall c \in C$

- $maxColor$ must account for the largest color index used:
  $maxColor \geq c \cdot y(c) \quad \forall c \in C$

4.4 Results

The results are obtained by minimizing the objective function, which represents the minimum bandwidth (i.e., the minimal range of colors needed to properly color the graph without conflicts).
This problem is the more complex to be solved because of the usage of the matrix decision variable x, as for the Graph Coloring problem, and the implementation of a fourth constraint to evaluate the $maxColor$.
We illustrate our results with a plot of a graph with 20 nodes and an edge probability of 0.3.

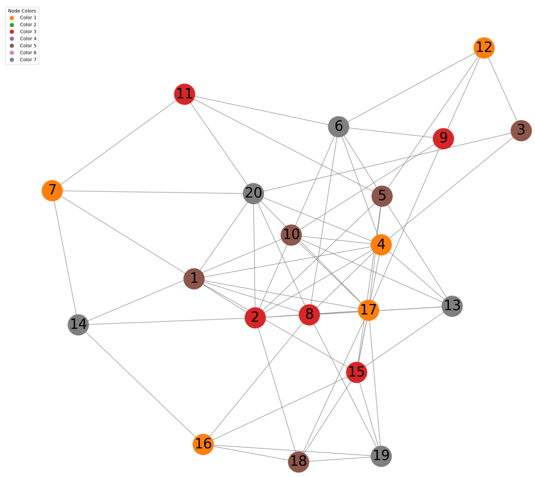

Figure 4: **Graph N=20 p=0.3**