**Assignment 4**

Advanced Algorithms

Gal Noy

June 2025

**Question 1**

**Section (a)**

Since the system is infeasible, by Farkas’ Lemma, there exists such that:

Consider the relaxed system . Multiply both sides by :

Since is a solution to a linear system with rational coefficients and polynomial-size input, we can assume each for some integer of polynomial size.

The number of inequalities is also at most polynomial in the input size, so:

Now choose large enough so that:

This ensures:

By Farkas’ Lemma, this implies that the system is also infeasible.

**Section (b)**

Since is feasible, there exists some point such that .

Define the ball:

Let . Then, for each row of ,

By Cauchy-Schwarz:

Assume each for some polynomial-size . Then:

Choose such that .

Then, for all :

Hence, the relaxed system is feasible and the entire ball is contained in it.

**Question 2**

**Max-Clique**

We define a binary variable for each vertex , where if the is included in the clique, and otherwise. The IP is:

Maximize:

Subject to:

The main constraint ensures that non-adjacent vertices cannot both be in the clique. This guarantees that the selected subset of vertices forms a complete subgraph.

The objective function counts the number of selected vertices, so maximizing it corresponds to finding a maximum-size clique.

**Subset-Sum**

We define a binary variable for each vertex , where if the is included in the subset, and otherwise. The IP is:

Maximize:

Subject to:

The constraints together enforce the equality , which ensures that the sum of the selected elements is exactly equal to the target .

The objective function is not essential, since we are only interested in whether such a subset exists, not how many elements are in it. Therefore, any feasible solution satisfies the problem.

**Min-Coloring**

Since at most colors are needed, we define:

* A binary variable for each color , where if the color is used, and otherwise.
* A binary variable for each vertex-color pair , where if vertex is assigned , and otherwise.

The IP is:

Minimize:

Subject to:

The first and the second constraints enforce the equality , which ensures each vertex is assigned exactly by one color.

The third constraint prevents adjacent vertices from being assigned the same color.

The objective function minimizes the total number of colors used by summing the binary variables , each of which indicates whether the color is active in the solution.

**Question 3**

**Notation**: for each , let denote the set of neighbours of in the graph.

We define the following algorithm:

* Initialize
* While :
  + Select a vertex , chosen uniformly at random from .
* Return .

**Correctness**

The returned set is independent, as each time a vertex is added to , all of its neighbors are immediately removed from the graph. This ensures no future vertex added to is adjacent to , and thus no two vertices in share an edge.

**Approximation Ratio**

We now show that this algorithm is an -approximation algorithm for the Maximum Independent Set problem.

Let denote the size of a maximum independent set.

In each iteration, the algorithm adds one vertex to , and removes and all its neighbours from the graph.

Since the degree of any vertex is at most , the number of removed vertices is at most:

Therefore, the number of iterations (and thus the size of ) is at least:

Since , we have:

Thus, the algorithm achieves a -approximation in expectation.

**Question 4**

**Notation**: for each , let denote the set of intersecting edges with in the hypergraph.

We define the following algorithm:

* Initialize
* While :
  + Select an edge , chosen uniformly at random from .
* Return .

**Correctness**

The returned set is a vertex cover: in each iteration, the algorithm selects an edge and add all of its vertices to . Then, it removes all edges that intersect , i.e., all edges that share at least one vertex with it. This ensures that each edge in the hypergraph is eventually removed only after at least one of its vertices has been added to , so every edge is covered.

**Approximation Ratio**

We now show that this algorithm is an 3-approximation algorithm for the vertex cover problem in 3-uniform hypergraphs.

Let denote an optimal, minimum-size vertex cover.

Let be the number of iterations of the algorithm. In each iteration, the algorithm covers one uncovered edge by adding all 3 of its vertices to , so:

Since each edge must be covered, and we remove all edges intersecting a chosen edge , any valid cover must contain at least one vertex from each such group. Thus, must have at least vertices, implying:

The algorithm constructs a valid vertex cover of size at most . Therefore, it is a 3-approximation algorithm for the vertex cover problem in 3-uniform hypergraphs