HW 9

Due: November 13th, 2025

Problem 1:

Semidefinite programming gives state-of-the-art approximation algorithms for essentially all problems where we are given a set of constraints, each constraining the allowed settings of a small number of variables, and we wish to find a setting of the variables that maximizes the number of satisfied constraints. As an example, we consider the MAX 2SAT problem: recall that in this problem, we are given a set of *clauses* on at most two *literals*. A literal is a Boolean variable or the negation of a Boolean variable, and a clause is just the Boolean OR: so, the MAX 2SAT constraints require that one of two variables take a specified truth value. Although it is easy to find an assignment to the variables that satisfies all of the clauses (when such an assignment exists), maximizing the number of satisfied clauses in an unsatisfiable formula is an NP-hard optimization problem.

Part A.

As in the case of the maximum cut problem, we'd like to express that MAX 2SAT problem as a "integer quadratic program" in which the only constraints are $y_i \in \{-1, 1\}$ and the objective function is quadratic in the y_i. Show that the MAX 2SAT problem can be expressed this way. (Hint: it may help to introduce a variable y_0 which indicates whether the value -1 or 1 represents "TRUE")

Part B.

Propose a 0.878-approximation algorithm for the MAX 2SAT problem. Prove that your algorithm obtains a 0.878-approximation and runs in polynomial time.

Problem 2:

This problem explores the analysis of an online caching algorithm using a linear programming (LP) primal-dual framework.

Background & Definitions

Recall that in modern computers, when we access a value in memory, a contiguous block of memory (that we will call a *page*) is loaded into some faster memory called a *cache*. Subsequent accesses to the page while it is stored in the cache are much faster. However, the size of the cache is limited; when enough distinct pages have been accessed, one of them must be removed from the cache (*"evicted"*) to make room for a new page to be loaded. Then, if we wish to read a value from a page that has been evicted, we must load it again. Since loading a page takes time, the objective in caching is to minimize the number of times pages are loaded while serving a sequence of requests to addresses in memory. We suppose the cache can hold k pages out of a "universe" of n total pages, and we receive a sequence of T page requests. Let $r(t) \in \{1, \ldots, n\}$ denote the page requested on the tth access, $t = 1, \ldots, T$.

We will use an LP relaxation to model the optimal *offline* solution (i.e., the optimal choice of evictions supposing that we know all of the requests $r(1), \ldots, r(T)$ in advance).

LP Variables

- x_i^t: A variable for each page i at time t. It is 1 if page i is absent from the cache immediately after serving the request at time t, and 0 otherwise.
- z_i^t: A variable for each page i at time t. It is 1 if we evict page i at time t, and 0 otherwise.

Primal LP Relaxation

The objective is to minimize the total number of evictions, which is equivalent to minimizing the number of times we load pages from memory.

$$
\begin{aligned}
\min \quad & \sum_{t=1}^{T} \sum_{i=1}^{n} z_i^t & \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i^t \geq n - k & \forall t \in \{1, \ldots, T\} \\
& x_{r(t)}^t \leq 0 & \forall t \in \{1, \ldots, T\} \\
& x_i^t \leq 1 & \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\} \\
& x_i^t - x_i^{t-1} \leq z_i^t & \forall i \in \{1, \ldots, n\}, t \in \{2, \ldots, T\} \\
& z_i^t \geq 0 & \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\} \\
& x_i^t \geq 0 & \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\}
\end{aligned}
$$

Part A.

Write the dual linear program of the primal LP provided above.

Hint: Other than the final non-negativity constraints (on the variables x_i^t and z_i^t), there are four types of constraints in the primal LP. These should give rise to four corresponding types of variables in the dual.

Part B.

Next, we will define an online algorithm for this problem.

Background: The 1-bit LRU Algorithm

The 1-bit LRU algorithm is an online paging algorithm. Each page i in the cache maintains a single "use bit," b_i, which is initialized to 0.

When a page $i = r(t)$ is requested:

1. **Cache Hit:** If page i is already in the cache, its use bit is set to 1 (i.e., $b_i \leftarrow 1$).
2. **Cache Miss (Page Fault):** If page i is not in the cache:
   (a) **If cache is not full:** Page i is added to the cache. Its bit is set to 1 ($b_i \leftarrow 1$).
   (b) **If cache is full:** The algorithm must evict a page.
      i. It scans the k pages in the cache looking for a page j with its use bit at 0 ($b_j = 0$). The first such page found is evicted.
      ii. If all k pages in the cache have their use bit set to 1, the algorithm first resets all use bits for all pages in the cache to 0 ($b_j \leftarrow 0$ for all j in cache). It then evicts an arbitrary page from this set.
      iii. After an eviction, the new page i is brought into the cache and its bit is set to 1 ($b_i \leftarrow 1$).

Task

Give a primal-dual analysis of the online 1-bit LRU algorithm, showing that it suffers at most $k \cdot (\mathrm{OPT} + 1)$ page faults.

Hint: One of the types of variables in your dual (from part a), let's call them d_i^t, should keep track of whether page i is "marked" at time t.

Define a potential function $\Phi(t) = \sum_i d_i^t$. Let $P(t)$ be the primal cost incurred by the 1-bit LRU algorithm at time t (i.e., $P(t) = 1$ if there is a page fault, $P(t) = 0$ otherwise) and let $D(t)$ be the increase to the dual objective at time t.

Show that you can maintain primal and dual feasible solutions online such that for all t:

$$P(t) \leq k \cdot D(t) + (\Phi(t) - \Phi(t-1))$$