10-725 Optimization, Spring 2010: Final Exam

Due: Tuesday, May 4th by noon

Instructions  There are 6 questions on this exam. Two questions involve coding. Do not attach your code to the writeup. Instead, copy your implementation to

/afs/andrew.cmu.edu/course/10/725/Submit/your_andrew_id/final

for andrew, or you can log into, for example, unix.andrew.cmu.edu. Please submit your solutions with your name and userid on the front page.

You may not use late days for the final exam, and you may not use Google or speak with other students about the questions. You can use any book listed on the course web site, our slides, and any other material on the course web site. Note that some problems are labeled as extra credit. As with all extra credit in this course, we will calculate letter grades *before* taking extra credit into account, and then use the extra credit only to increase your grade.

1  Dual cone and conjugate function [Yi, 5 points]

Let $K$ be a cone and $K^* = \{ y \mid x^T y \geq 0 \text{ for all } x \in K \}$ be its dual cone. Let’s define a function $f(x) = I_K(x)$, i.e., the indicator function taking value 0 if $x \in K$ and value $+\infty$ otherwise. Derive the conjugate (i.e., dual) $f^*(y)$ of the function $f(x)$, and express it using $K^*$. Note: use the notation $-K^* = \{ y \mid -y \in K^* \}$.

2  Minimizing Euclidean distance with $\ell_1$ constraints [Sivaraman, 20 points]

Consider the following $\ell_1$ constrained minimization problem

$$\begin{align*}
\min_x & \quad \frac{1}{2} ||x - x_0||_2^2 \\
\text{s.t.} & \quad ||x||_1 \leq z
\end{align*} \tag{1}$$

with variables $x \in \mathbb{R}^n$. Here we want to minimize the Euclidean distance to a given point $x_0 \in \mathbb{R}^n$ with constrained $\ell_1$ norm on our variables. Assume $z > 0$ is given. This problem is very useful as it is usually the building block of many optimization problems with $\ell_1$ norm constraints, e.g., projected subgradient with $\ell_1$ norm constraints. We will propose an efficient method to solve this problem.

2.1  Reduction of the $\ell_1$ constraint into the simplex constraint

The first step to solve our problem is to reduce problem (1) to another easier problem of the following form:

$$\begin{align*}
\min_w & \quad \frac{1}{2} ||w - v||_2^2 \\
\text{s.t.} & \quad \sum_{i=1}^n w_i = z \\
& \quad w_i \geq 0, \forall i
\end{align*} \tag{2}$$

with variables $w \in \mathbb{R}^n$. 

1
1. [1 pts] In the case that $||x_0||_1 \leq z$, what is the optimal $x^*$ to the original problem?

2. [1 pts] If $||x_0||_1 > z$, what can we say about the constraint of the original problem (i.e., is that still an inequality constraint, or can we convert it to an equality constraint? Why?). Re-write the original problem using this fact.

3. [1 pts] In the rewritten problem, what is the relationship between the signs of (the elements of) $x$ and $x_0$.

4. [2 pts] Using the facts above, show a reduction from (1) to (2) for the case $||x_0||_1 > z$. (Hint: how would you construct (2) given (1)? Also, show how you would use the solution to (2) to obtain the solution to (1)).

### 2.2 Characterizing the optimal solution with the simplex constraint

Now we need to solve problem (2). We start this by characterizing the optimal solution to the problem. In this part, without loss of generality, let’s assume the elements of $v$ are in descending order, i.e., $v_1 \geq v_2 \geq \ldots \geq v_n$.

1. [2 pts] Prove the following lemma: let $w^*$ be an optimal solution to problem (2), and let $s$ and $j$ be two indices such that $v_s > v_j$. Then if $w^*_s = 0$, we must have $w^*_j = 0$ as well.

2. [2 pts] Prove the following lemma: let $w^*$ be an optimal solution to problem (2), and let $s$ and $j$ be two indices such that $v_s = v_j$. Then we must have $w^*_s = w^*_j$.

3. [1 pts] Prove the following lemma: let $w^*$ be an optimal solution to problem (2) and let $I$ be the indices of strictly positive elements of $w^*$, i.e., $I = \{i : w^*_i > 0\}$. Then, $I$ must be in the form of $\{1, 2, 3, \ldots, k\}$ for certain $k \leq n$. In other words, all positive elements of $w^*$ are located consecutively from the beginning.

### 2.3 Solving the problem

Now we start to solve problem (2). Note that we still assume that the elements of $v$ are in descending order so that the lemmas we just proved are valid. Now suppose we know $k$, the number of positive elements in the optimal solution $w^*$, from an oracle\(^2\). Now we know the first $k$ coefficients of $w^*$ are positive, and others are zeros. We will derive an efficient method to find those $k$ positive coefficients of $w^*$.

1. [2 pts] Write the Lagrangian of the problem (2), using $\theta$ to denote the free Lagrange multiplier and $\{\lambda_i\}_{i=1}^n$ to denote the set of non-negative Lagrange multipliers.

2. [2 pts] Write the KKT conditions for this problem.

3. [2 pts] Using complementary slackness, further write out expressions characterizing the relationship among $\theta$, $v$, and nonzero elements of $w$. More specifically, write an expression of $w_i$ using $v_i$, $\theta$ and $\lambda_i$. Then eliminate $\lambda_i$ using the fact that $w_i > 0$ for these nonzero elements.

4. [2 pts] Using $z$ given in the problem and $k$ given by the oracle, write out an expression by which we can solve for the Lagrange multiplier $\theta$. Hint: Relate $\sum_{i=1}^k(v_i - \theta)$ to $z$ using $\{w_i\}$. Then solve for $\theta$ using the fact that $v$ and $z$ are known.

5. [2 pts] Solve for the optimal solution $w^*$ in terms of $v$ and $\theta$.

---

\(^1\)If not, we can re-arrange the elements of $w$ and $v$ to get an equivalent problem, where the elements of $v$ are in descending order.

\(^2\)Proof of the existence and the closed form of this oracle is a little bit involved, so we don’t require it here.
3 Facility location and LP relaxation [Yi, 25 points]

In this question, we study facility location and its linear programming relaxation. The problem\(^3\) studies design situations for opening a set of facilities. The data given to the problem are:

- A set of potential facilities \(F = \{1, 2, 3, \ldots, |F|\}\).
- A set of cities \(C = \{1, 2, 3, \ldots, |C|\}\).
- Costs of opening facilities \(\{f_i > 0\}_{i \in F}\).
- Costs of connecting cities to facilities \(\{c_{ij} \geq 0\}_{i \in F, j \in C}\), which are guaranteed to be a metric and thus satisfy the triangle inequality. (For example, the costs might be the Euclidean distances between cities and facilities.)

Given the data, we want to find out:

- A subset \(I \subseteq F\), which indicates facilities to open.
- An assignment function \(\phi(j) : C \rightarrow I\), which assigns each city \(j \in C\) to an opened facility \(\phi(j) \in I\).

The goal is to minimize the total cost:

\[
\sum_{j \in C} c_{\phi(j),j} + \sum_{i \in I} f_i \tag{3}
\]

We can write this problem as an integer programming using the following variables: 1) opening decisions of facilities \(\{y_i\}_{i \in F}\), which take value 1 if \(i \in I\) and 0 otherwise; 2) connecting decisions \(\{x_{ij}\}_{i \in F, j \in C}\), which take value 1 if \(\phi(j) = i\) and 0 otherwise. The integer programming problem is

\[
\begin{align*}
\min & \quad \sum_{i \in F, j \in C} c_{ij}x_{ij} + \sum_{i \in F} f_i y_i \\
\text{s.t.} & \quad \sum_{i \in F} x_{ij} \geq 1 \quad \forall j \in C \\
& \quad y_i - x_{ij} \geq 0 \quad \forall i \in F, \forall j \in C \\
& \quad x_{ij} \in \{0, 1\} \quad \forall i \in F, \forall j \in C \\
& \quad y_i \in \{0, 1\} \quad \forall i \in F
\end{align*} \tag{4}
\]

Note that the first set of constraints ensures that each city is connected to at least one facility, and the second set of constraints ensures that the connected facilities must be opened.

3.1 LP Relaxation, Duality and Complementary Slackness

The above integer programming is NP-hard. A simple relaxation to the problem is to treat binary variables as continuous, and the resulting problem is an LP. (Note that we have dropped some constraints like \(y_i \leq 1\); you may want to convince yourself that these constraints can never influence the solution.)

\[
\begin{align*}
\min & \quad \sum_{i \in F, j \in C} c_{ij}x_{ij} + \sum_{i \in F} f_i y_i \\
\text{s.t.} & \quad \sum_{i \in F} x_{ij} \geq 1 \quad \forall j \in C \\
& \quad y_i - x_{ij} \geq 0 \quad \forall i \in F, \forall j \in C \\
& \quad x_{ij} \geq 0 \quad \forall i \in F, \forall j \in C \\
& \quad y_i \geq 0 \quad \forall i \in F
\end{align*} \tag{5}
\]

1. [5 pts] Use the Lagrangian and KKT conditions to derive the dual problem of the LP (5). To make our grading easier, please use \(\{\alpha_j\}_{j \in C}, \{\beta_{ij}\}_{i \in F, j \in C}, \{\gamma_{ij}\}_{i \in F, j \in C}, \{\lambda_i\}_{i \in F}\) to denote the Lagrange multipliers for the four sets of constraints in (5), respectively. Note that you can eliminate \(\{\gamma_{ij}\}\) and \(\{\lambda_i\}\) in your final dual form by treating them as slack variables.

2. [1 pts] Suppose we solve the primal LP and obtain an optimal solution \(\{x_{ij}^*, y_i^*\}\). The corresponding dual optimal solution contains Lagrange multipliers \(\{\alpha_j^*\}, \{\beta_{ij}^*\}\). Based on complementary slackness, what can we say about \(\{x_{ij}^*\}_{i \in F}\) if \(\alpha_j^* > 0\)? (fact 1)

\(^3\)This is a different version than the one we knew from the class.
Algorithm 1 4-approximation LP rounding algorithm

**Input:** facility costs \( \{f_i\} \), connecting costs \( \{c_{ij}\} \), optimal primal LP solution \( \{x_{ij}^*\}, \{y_i^*\} \), and optimal dual multipliers \( \{\alpha_{ij}^*\}, \{\beta_{ij}^*\} \) for the first two groups of constraints.

**Output:** a set of opened facilities \( I \subseteq F \), and an assignment function \( \phi(j) : C \rightarrow I \).

Initialize: \( I \leftarrow \emptyset \)

Initialize: \( \phi(j) \leftarrow \text{unassigned} \), \( \forall j \in C \)

Define: \( N_j = \{i \in F : x_{ij}^* > 0\}, \forall j \in C \) (i.e., the facilities connected to city \( j \) according to primal LP).

repeat

Find an unassigned city \( j \) with smallest \( \alpha_{ij}^* \).

Open the cheapest facility \( i \) in \( N_j \), i.e., \( I \leftarrow I \cup \{\arg\min_{i \in N_j} f_i\} \). (Break ties arbitrarily.)

Assign every unassigned city \( j' \) s.t. \( N_{j'} \cap N_j \neq \emptyset \) to the facility \( i \) (note: city \( j \) included).

until no city is unassigned

3. [1 pts] Based on complementary slackness, what can we say about \( y_i^* \) and \( x_{ij}^* \) if \( \beta_{ij}^* > 0 \)? (fact 2)

4. [2 pts] Based on KKT conditions (both complementary slackness and gradient conditions), what can we say about \( \alpha_{ij}^* \), \( \beta_{ij}^* \) and \( c_{ij} \) if \( x_{ij}^* > 0 \)? (fact 3)

5. [2 pts] Based on KKT conditions (both complementary slackness and gradient conditions), what can we say about \( f_i \) and \( \{\beta_{ij}^*\}_{j \in C} \) if \( y_i^* > 0 \)? (fact 4)

3.2 An Intuitive Understanding of the Dual LP

Give the dual variables \( \{\alpha_{ij}\}_{j \in C}, \{\beta_{ij}\}_{i \in F,j \in C} \) the following intuitive meanings: \( \beta_{ij} \) is the amount of money city \( j \) is willing to pay for opening facility \( i \), and \( \alpha_{ij} \) is the total amount of money city \( j \) is willing to pay (both for opening facilities and for creating connections). Now we have an interesting intuition for the primal and dual LPs: the primal problem tries to minimize the total cost of the solution, while the dual problem tries to maximize the total money all cities are willing to pay.

1. [2 pts] Give intuitive explanations for the constraints in your dual LP (note: no need to explain non-negativity constraints, and dual variables \( \gamma_{ij} \) and \( \lambda_i \) should already be eliminated in your dual). Use at most one sentence for each set of constraints.

2. [2 pts] Give intuitive explanations for fact 3 and fact 4 we developed in section 3.1. Use one sentence for each fact if possible.

3.3 A 4-Approximation LP Rounding Algorithm

Suppose we solve both primal and dual LPs, and obtain the optimal solutions \( \{x_{ij}^*\}, \{y_i^*\}, \{\alpha_{ij}^*\}, \{\beta_{ij}^*\} \). Generally, \( \{x_{ij}^*\}, \{y_i^*\} \) are not integral and thus not valid for the original problem. Based on LP solutions, we use Algorithm 1 to get a valid solution. The algorithm starts with no facility opened and no city assigned. In each iteration, the algorithm picks an unassigned city \( j \) with smallest \( \alpha_{ij}^* \), and opens the cheapest facility \( i \) in \( N_j \). (Here, \( N_j \) is the set of facilities \( i \) that are neighbors to city \( j \) in the primal solution, i.e., where \( x_{ij}^* > 0 \).) Then, the algorithm assigns every unassigned city \( j' \) s.t. \( N_{j'} \cap N_j \neq \emptyset \) to the newly opened facility \( i \). The algorithm iterates until all cities have been assigned to some opened facility.

The algorithm returns the set of opened facilities \( I \subseteq F \) and the assignment function \( \phi(j) : C \rightarrow I \). The final total cost for this rounded solution, as defined in eq. (3), is:

\[
V(I, \phi) = \sum_{j \in C} c_{\phi(j),j} x_{ij}^* + \sum_{i \in I} f_i.
\]

Let \( OPT(IP) \) denotes the value of the true optimal solution to the original integer program (4). We want to prove that \( V(I, \phi) \leq 4 \cdot OPT(IP) \), i.e., the LP rounding algorithm provides a 4-approximation.

Let \( OPT(P) \) and \( OPT(D) \) denote the optimal values of the primal LP with \( \{x_{ij}^*\}, \{y_i^*\} \) and dual LP with \( \{\alpha_{ij}^*\}, \{\beta_{ij}^*\} \), respectively. We know that \( OPT(D) = OPT(P) \leq OPT(IP) \) due to strong duality in
LPs. As a result, we can split \( V(I, \phi) \) into two parts and prove that: 1) \( \sum_{i \in I} f_i \leq \text{OPT}(P) \), and 2) \( \sum_{j \in C} c_{\phi(j)} \leq 3 \cdot \text{OPT}(D) \). If we can do so, we obviously get \( V(I, \phi) \leq \text{OPT}(P) + 3 \cdot \text{OPT}(D) \leq 4 \cdot \text{OPT}(IP) \).

1. [5 pts] Prove that \( \sum_{i \in I} f_i \leq \text{OPT}(P) \), where \( I \) is returned by Algorithm 1 and \( \text{OPT}(P) \) is the optimal value of primal LP (5) with optimal solution \( \{x^*_i\}, \{y^*_j\} \)?

   **Step 1** [1 pt]: argue that proving \( \sum_{i \in I} f_i \leq \sum_{i \in F} f_i y^*_i \) is sufficient (as \( \sum_{i \in F} f_i y^*_i \) is part of \( \text{OPT}(P) \)).

   **Step 2** [3 pts]: Each element of \( I \) is returned by one iteration of Algorithm 1. Consider one iteration, which starts with an unassigned city \( j \) and opens the cheapest facility \( i = \arg \min_{i \in N_j} f_i \) in \( N_j \). The cost of this facility is thus \( \min_{i \in N_j} (f_i) \), which is exactly the contribution of this iteration to the left-hand side of the inequality in step 1. For such an iteration, try to establish the inequality

\[
\min_{i \in N_j} (f_i) \cdot 1 \leq \min_{i \in N_j} (f_i) \cdot \sum_{i \in N_j} x^*_{ij} \leq \min_{i \in N_j} (f_i) \cdot \sum_{i \in N_j} y^*_i \leq \sum_{i \in N_j} f_i y^*_i
\]

Note that KKT conditions for the LP optimal solution \( \{x^*_i\}, \{y^*_j\} \) (especially the primal constraints) as well as the definition of \( N_j \) are useful to establish the above inequality.

**Step 3** [1 pt]: In step 2 we develop an inequality \( \min_{i \in N_j} (f_i) \cdot 1 \leq \sum_{i \in N_j} f_i y^*_i \) for each iteration of the algorithm. Argue how we can combine these inequalities to prove the overall inequality that we decided we needed in step 1.

2. [5 pts] Prove that \( \sum_{j' \in C} c_{\phi(j'),j'} \leq 3 \cdot \text{OPT}(D) \). (Note: we use \( j' \) instead of \( j \) for notational convenience).

   General idea: \( \sum_{j' \in C} c_{\phi(j'),j'} \) is the sum of the connection costs of cities which are assigned to a facility by the algorithm. Consider one such city \( j' \in C \), assigned by the algorithm at a certain iteration. This iteration must start with picking an unassigned city \( j \) and opening a facility \( i = \arg \min_{i \in N_j} f_i \), i.e., the cheapest facility from \( N_j \). In this iteration, city \( j' \) is assigned to facility \( i \) because \( N_{j'} \cap N_i \neq \emptyset \). The contribution of such an assignment \( \phi(j') = i \) to \( \sum_{j' \in C} c_{\phi(j'),j'} \) is exactly \( c_{ij'} \). The following steps help to find an upper bound for each \( c_{ij'} \) and finally prove the theorem.

   **Step 1** [3 pts]: Establish an inequality \( c_{ij'} \leq 2\alpha^*_i + \alpha^*_j \). Note that \( j' \) is chosen because \( N_{j'} \cap N_i \neq \emptyset \) for the city \( j \) picked at the beginning of the iteration. In other words, \( x^*_{ij'} > 0 \) and \( x^*_{ij'} > 0 \) for some shared facility \( i' \). By the triangle inequality, \( c_{ij'} \leq c_{ij} + c_{ij'} + c_{ij'} \) because of the path \( i \to j \to i' \to j' \), where \( i, j, j' \) are all defined above. So, we need to prove \( c_{ij} + c_{ij'} + c_{ij'} \leq \alpha^*_i + \alpha^*_j + \alpha^*_j \). For this, use our knowledge about \( x^*_{ij}, x^*_{ij'}, x^*_{ij'} \) in Algorithm 1, and complementary slackness of primal and dual optimal \( \{x^*_i\}, \{y^*_j\}, \{\alpha^*_i\}, \{\beta^*_j\} \) (Section 3.1).

   **Step 2** [1 pt]: Prove \( c_{ij'} \leq 3 \cdot \alpha^*_i \) using the result in step 1.

   **Step 3** [1 pt]: Prove \( \sum_{j' \in C} c_{\phi(j'),j'} \leq 3 \cdot \text{OPT}(D) \) using the result in step 2.

4 Submodular functions [Sivaraman, 10 points]

1. [2 pts] Show that a function \( f(A) = g(|A|) \) is submodular if and only if \( g \) is concave, where \( g \) is a function mapping a natural number to a real number.

2. [2 pts] Show that the function \( f(S) = \sum_{i \in S} w_i \) is modular for any set of non-negative weights.

3. [2 pts] Show that the function \( f(S) \), defined as the rank of the subspace spanned by the set of vectors \( S \), is submodular.

4. [4 pts] Consider a weighted undirected graph \( G \) with vertices \( V \), edges \( E \), and nonnegative weights \( w_{ij} \geq 0 \) for \( (i,j) \in E \). A cut in this graph separates a set of vertices \( A \) from the rest of the vertices \( V \setminus A \). The weight of this cut is the sum of the weights of the edges that cross the cut:

\[
cut(A) = \sum_{(i,j) \in E : i \in A \land j \in V \setminus A} w_{ij}.
\]
• [1 pts] Is cut(A) a symmetric function? Justify.
• [2 pts] Prove that cut(A) is a submodular function.
• [1 pts] What do these results imply about the complexity of the minimum cut problem? What would be the running time of applying Queyranne’s algorithm for minimizing submodular functions to the minimum cut problem?

5 Branch and Bound for MAX-SAT [Yi, 25 + 5 points]

The goal of this problem is to illustrate the technique of branch and bound, and use it to obtain the optimal solution to an instance of MAX-3SAT.

MAX-SAT: Given a conjunctive normal form (CNF) formula $f$ on Boolean variables $x_1, x_2, \ldots, x_n$, and non-negative weights $w_c$ for each clause $c$ of $f$, find an assignment to the variables that maximizes the total weight of satisfied clauses.

Example: Let $f = c_1 \land c_2 \land c_3$, where $c_1 = x_1 \lor \overline{x_2}$, $c_2 = x_2$, $c_3 = \overline{x_1} \lor \overline{x_2}$, $w_{c_1} = 2$, $w_{c_2} = 3$, $w_{c_3} = 4$. (Here the variables are $x_1, x_2$, and $\overline{x_i}$ denotes negation.) In this case, the assignment that achieves the maximum weight is $x_1 = 0$ (false), and $x_2 = 1$ (true). Clauses $c_2$ and $c_3$ are satisfied, and $c_1$ is not.

In this question, we will restrict each clause to have at most three literals. This version of the MAX-SAT problem is known as MAX-3SAT. The decision version of the problem, i.e., deciding whether the maximum value is greater than or equal to some number, is NP-hard. (Note that this definition is slightly different from the one we used class, where MAX-3SAT meant that each clause had exactly three distinct literals.)

Let us define the shorthand $x_{n+i} = \overline{x_i}$, where $n$ is the total number of variables. Let us also define an additional variable $x_0$ which is equal to 0. With these definitions, a clause $c = x_i \lor x_j$ containing just two literals can be expressed as $c = x_0 \lor x_i \lor x_j$ and a clause containing a single literal $c = x_i$ can be expressed as $c = x_i \lor x_0 \lor \overline{x_i}$. Moreover, if a variable $x_i$ occurs negated in a clause, we can replace it by the variable $x_{n+i}$; e.g., $c = \overline{x_i} \lor x_j \lor \overline{x_k} = x_{n+i} \lor x_j \lor x_{n+k}$. Thus, we can assume w.l.o.g. that all of our clauses are of the form $x_i \lor x_j \lor x_k$, where $0 \leq i, j, k \leq 2n$ (although we may have some duplicated indices, e.g., $i = j$). Write $\langle i, j, k \rangle$ for the clause $x_i \lor x_j \lor x_k$, and write $w_{ijk}$ for the weight of clause $\langle i, j, k \rangle$.

Data: The data for this question consists of the matlab file graph_plan.m. The file expresses a propositional planning problem as a maximum satisfiability problem, so that the cost and feasibility of a plan are related to the weight of the unsatisfied clauses. The file defines a matrix, each row of which represents a clause in the CNF formula: we represent a clause $\langle i, j, k \rangle$ as a row containing the elements $i, j, k$. For this problem, the number of variables is $n = 48$. The file also contains a cost vector that defines the weight for each clause, as well as code for solving the LP relaxation for the planning problem. We describe the relaxation below.

Let us first obtain an exact Integer Program for the MAX-3SAT problem. Let $\mathcal{C}$ and $\mathcal{V}$ denote the set of clauses and the set of variables respectively. We define a variable $\zeta_{ijk}$ for each clause $\langle i, j, k \rangle$, that takes the value $+1$ if the clause is unsatisfied, and takes the value $0$ if the clause is satisfied. Note that maximizing the weight of satisfied clauses is the same as minimizing the weight of unsatisfied clauses. Thus our IP problem is

$$\min \sum_{i,j,k} w_{ijk} \zeta_{ijk}$$

subject to

$$x_i + x_j + x_k + \zeta_{ijk} \geq 1 \quad \forall \langle i, j, k \rangle \in \mathcal{C}$$
$$x_i + x_{n+i} = 1 \quad \forall i \in \{1, \ldots, n\}$$
$$x_i \in \{0, 1\} \quad \forall i \in \{1, \ldots, 2n\}$$
$$\zeta_{ijk} \in \{0, 1\} \quad \forall \langle i, j, k \rangle \in \mathcal{C}$$

4See Part IV of ‘Artificial Intelligence: A Modern Approach’ by Russell and Norvig if you are interested in knowing about how to express a planning problem in CNF. This is not needed for the purpose of this question.
The LP relaxation of this IP is obtained by removing the last two integrality constraints from the above formulation. We replace these constraints by the constraints $0 \leq x_i \leq 1$ and $0 \leq \zeta_{ijk} \leq 1$.

1. [10 pts] Using the above LP relaxation (graph_plan.m), implement a branch and bound algorithm that attempts to compute the optimal value of the MAX-3SAT problem defined above. For doing branch and bound, we need a heuristic to choose which variable to branch on, and which branch (i.e., which value of the variable) should we try first. One way to do this is to branch on the variable that we are most sure about, i.e., choose a variable that takes an integral value or a value most close to an integer. In this case, we first try setting the variable to the integer value being suggested by the LP relaxation. Another way is to choose a variable we are least sure about, i.e., whose value is closest to 0.5. In this case, we can pick either value to try out first. You can use either of these heuristics, or come up with your own heuristics. Describe the heuristic you use to do branch and bound, including the criteria you use to prune your search.

2. [5 pts] Report the values of the variables and the set of unsatisfied clauses, in a table. Also include it in your submission directory as a text file having the following format: Each line is of the form “$i, x_i$”, which reports the value of the $i$-th variable. Once the values of all variables have been specified, there is a blank line. After this, each line is of the form “$i$”, which denotes the index of an unsatisfied clause.

3. [10 pts] At each node of the depth first search tree, obtain an integer solution from the LP relaxation by randomized rounding, i.e., if a variable $x_i$ has the value $v$, then round it to 1 with probability $v$, and round it to 0 with probability $1 - v$. Do this rounding independently for each $x_i$, $i \in \{1, \ldots, 48\}$. Set the remaining variables $x_{48+i} = 1 - x_i$, $i \in \{1, \ldots, 48\}$. Now check whether each clause is satisfied or not, and set the slack variable corresponding to that clause to 0 if the clause is satisfied, and 1 otherwise. Argue that the solution obtained using this procedure is integral and is a feasible solution to the original Integer Program. Therefore, each such integer solution provides an upper bound on the optimal integer solution. At every point of the depth first exploration, maintain the best value (i.e., the lowest value) for the integer solution seen thus far. Plot this value as a function of running time, and also as a function of number of nodes explored, for 3 different runs of the branch and bound procedure. These plots show the convergence behavior of our branch and bound algorithm.

4. [extra 5 pts] Branch and cut: We now modify the branch and bound technique above. At each step, before solving the LP relaxation, introduce a cut, i.e., add additional constraints to the LP.

One possible heuristic is as follows. Resolution: Suppose we have 2 clauses of the form $w \lor \bar{y} \lor z$ and $x \lor y \lor z$. Satisfying them simultaneously indicates satisfying the clause $w \lor x \lor z$. This technique of combining two clauses containing a variable and its negation is known as resolution. Notice that if the two clauses do not have any common variables other than the variable that is eliminated, then combining the two clauses is equivalent to adding the corresponding constraints in the LP formulation / LP relaxation. Thus, resolution does not yield an additional non-trivial constraint. If, however, the clauses share variables other than the ones being eliminated ($z$ in the example above), then doing resolution is not equivalent to adding the corresponding constraints. This is because instead of having 2 copies of the variable $z$, it suffices to keep just a single copy. In terms of constraints, this is equivalent to summing the constraints, and rounding down the coefficient of the variable $z$ from 2 to 1. This is an additional constraint and potentially cuts down / reduces the size of the feasible set for the LP relaxation. Note that the discussions here neglect the slack variables $\zeta_{ijk}$, but for MAT-SAT problems, we have to do with the slacks for the two clauses when we add the resolved clause.

Use resolution or some other heuristic to introduce additional constraints in the LP relaxation. If you use something other than resolution, describe it. Compare the running time and the number of nodes explored using this technique, with the branch and bound technique above. Obtain an integer feasible solution at each node of the DFS tree using the technique described in part 3 above, and plot the upper bound as a function of running time and as a function of the number of nodes explored. On the same plots, plot the corresponding values for the branch and bound technique.
Here are some additional discussions for branch and cut (in case that you like to learn something beyond the scope of the class). A general approach for the cut (not only specifically for MAX-SAT but also for other problems) is the Gomory cut\(^5\). The basic idea of a Gomory cut is that we take a linear combination of the constraints, round the coefficients of integer variables in the direction that makes the constraint looser (and therefore still valid), and then round the constant in the direction that makes the constraint tighter. This last step is valid since we know that an integer linear combination of integer variables is an integer, so, e.g., any integer \(\geq 4.2\) is also \(\geq 5\). This tighter constraint will not cut any integer solution, but it may help to reduce the feasible region of the LP relaxation. Depending on which linear combination we start with, the end result may be either trivial (looser than the original linear combination) or not (cuts off some non-integer solutions), depending on whether the loosening in the first step dominates the tightening in the second.

An example follows: the clauses \(x \lor y \lor z\) and \(y \lor a \lor \overline{z}\) can be represented as

\[
\begin{align*}
x + y + z + s_1 & \geq 1 \\
(1 - z) + y + a + s_2 & \geq 1 \\
x, y, z, a, s_1, s_2 & \in \{0, 1\}
\end{align*}
\]

(The slack variables \(s_1\) and \(s_2\) will be penalized in the objective.) Adding together these two inequalities yields

\[
x + 2y + a + s_1 + s_2 \geq 1
\]

We would like to get rid of the coefficient 2, changing 2\(y\) into \(y\). Without the slack variables, this would be analogous to logical resolution, in which we would conclude \(x + y + a \geq 1\).

To make a Gomory cut, we start with a linear combination of the two original inequalities (weight 0.5 on each):

\[
y + \frac{1}{2}x + \frac{1}{2}a + \frac{1}{2}s_1 + \frac{1}{2}s_2 \geq \frac{1}{2}
\]

In the first stage of the Gomory cut, we take the ceiling of all non-integer coefficients on the LHS; since all variables are nonnegative, this operation can only increase the LHS, leading to the inequality

\[
y + x + a + s_1 + s_2 \geq \frac{1}{2}
\]

In the second stage of the cut, we note that the LHS is an integer (since it is an integer combination of integer variables). All integers which are greater than or equal to \(\frac{1}{2}\) are also greater than or equal to 1; so, we get the valid cut

\[
y + x + a + s_1 + s_2 \geq 1
\]

as desired.

6 Logistic Lasso [Sivaraman, 30 points]

In this question you will implement a barrier method to solve the multi-class logistic regression problem with an \(L_1\) penalty on the weights.

6.1 Preliminaries

1. [1 pt] Suppose you are given a training set \((X, Y) = \{(X_1, y_1), \ldots, (X_n, y_n)\}\). Write an expression for the log-likelihood of the training set, and cast the problem of maximizing this log-likelihood with an \(L_1\) penalty of the form \(\lambda \sum_{i=1}^{K} |w_i|\) on the parameters as an optimization problem.

\(^5\)See Geoff’s advanced AI lecture if you are interested: http://www.cs.cmu.edu/~ggordon/780/slides/12-duality.pdf
2. [2 pts] Re-cast the above optimization problem as an inequality constrained optimization with a *smooth* objective.

*Hint: Introduce a new variable for each weight and re-write the $L_1$ penalty in terms of a set of linear inequalities*

3. [7 pts] You can now solve this optimization problem using the log-barrier method. Introduce a barrier function $\phi$, and rewrite the optimization problem as a smooth unconstrained optimization problem (given a fixed value for the barrier parameter $t$). Derive the gradient and Hessian of this objective (you don’t need to redo the work from HW-5, but you do need to take care of the new terms and new variables). Describe the barrier method in the context of solving this problem (you do need a line search for this problem).

### 6.2 Implementation

1. [10 pts] Implement the barrier method you described above. For the line search, use parameters $\alpha = 0.15$ and $\beta = 0.5$. Initialize $t = 1$, and to update $t$, use $t := \mu t$ with $\mu = 10$. Do 10 outer iterations (i.e. you should call your Newton’s subroutine 10 times). You can reuse any code from HW-5. You can also use our solution (which we will upload as “newton_lr.m”), although you are encouraged to use your own implementation since it will be easier to modify.

2. [3 pts] Now, you will rerun most of your experiments from HW-5. Again, feel free to reuse any code from the previous HW. Download the synthetic dataset (“data_large.mat”) from the website. Train your logistic regressor on the training set, and use the learned weights to predict class labels for the test set. In your report you should show the weights you learned (you can print these out since this is a big vector or matrix but clearly indicate the class and feature for the weights). Report your accuracy on the test set. Use $\lambda = 1$ for this part.

3. [3 pts] Download “data_small.mat” from the website. Plot regularization paths for the weights (you should have only 4 weights), by varying $\lambda$ (values of $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$ work well for me but you can use any values as long as you get a reasonable curve).

4. [4 pts] Download the UCI Iris dataset from the website (“iris_data.mat”). We have split the data for you into training, validation and test sets. Use the training and validation sets to pick the best value of $\lambda$ (i.e., the one which leads to the best accuracy on the validation set when we train on the training set). Please provide a plot to justify this choice of $\lambda$. Then fit a logistic regression on the combined training and validation sets, using only the best value of $\lambda$. In your report, clearly write down the selected value of $\lambda$ and the accuracy of the final regression on the combined training+validation sets and on the test set.