SA405 Test 1 Review

Key Definitions

1. **Basic Solution**: Let *S* be a polyhedral set defined by linear inequality and equality constraints. Solution **x** is a basic solution if (a) **x** satisfies all equality constraints of *S* and (b) at least *n* of the constraints of *S* are active at **x** and are linearly independent.
2. **Basic Feasible Solution**: If **x** also satisfies all constraints of *S*, then **x** is a basic feasible solution.
3. **Canonical Form of Linear Programs**:
4. To convert an LP into canonical form:
   1. For each variable that is unrestricted in sign, add the equality constraint and the inequalities and .
   2. For each variable , replace with , where
   3. Make sure RHS is non-negative
   4. Add logical variables: GREATER THAN means -Slack, LESS THAN means +Slack, if already an equality LEAVE IT ALONE
5. **Basic Variable**: Given a basic feasible solution **x** of a linear program in canonical form, a variable *xk* is called basic if it is one of the *m* linearly independent columns of A defining **x**.
6. **Nonbasic Variable**: A variable *xk* is nonbasic if it is not basic.
7. **Basis**: The set of basic variables is referred to as the basis of **x**.
8. **Convex Set**: A set *S* is convex if, for all , the points for all In simple terms, given two solutions in *S*, all solutions on the line segment connecting them are also in *S*.
9. **Extreme Point**: Given a convex set *S*, a solution is an extreme point if there does not exist two distinct solutions such that *x* is on the line segment joining *y* and *z*; that is, there does not exist a such that .

Types of Constraints

**Networks - Balance of Flow Constraints**

Let *V* denote the set of nodes and *A* denote the set of arcs in the network *G*. At each node , let denote the supply at node *i*, and the demand at node *j*, and define ; we assume that

which means the total amount of supply in the network equals the total demand. Then the balance of flow constraints are given by

**Fixed Charge General Model**

Consider the linear function , where each coefficient and each variable . We want to model whether the function has a positive value. We define to be

Then we can model this as

For some large number *M*, which we refer to as “Big-*M*”.

**Set Covering Models**

Given a collection *C* of choices, where we define variable

a covering constraint (select at least one) is of the form

a packing constraint (select at most one) is of the form

and a partitioning constraint (select exactly one) is of the form

**Either-Or Constraints**

Suppose we have the two constraints

and assume that at least one of these must hold. Letting *y* be a binary variable, we can define the two constraints

and make both constraints be satisfied.

**If-Then Constraints**

Suppose we want to model the logical situation: If , then . We can model this using the constraint

**Anti-Cycling Constraints**

MST Version

TSP Version