NC STATE UNIVERSITY

Operations Research Master's Oral Exam

Eric Sullivan

Solving the Max-Cut Problem using Semidefinite Optimization in a Cutting Plane Algorithm.
(under the direction of Dr. Kartik Sivaramakrishnan)

Wednesday, May 7, 2008
1:00 pm HA222

Abstract:

A central graph theory problem that occurs in experimental physics, circuit layout, and computational linear algebra is the max-cut problem. The max-cut problem is to find a bipartition of the vertex set of a graph with the objective to maximize the number of edges between the two partitions. The problem is NP-hard, i.e., there is no efficient algorithm to solve the max-cut problem to optimality.

We propose a semidefinite programming based cutting plane algorithm to solve the max-cut problem to optimality in this thesis. Semidefinite programming (SDP) is a convex optimization problem, where the variables are symmetric matrices. An SDP has a linear objective function, linear constraints, and also convex constraints requiring the matrices to be positive semidefinite. Interior point methods can efficiently solve SDPs and several software implementations like SDPT3 are currently available.

Each iteration of our cutting plane algorithm has the following features: (a) an SDP relaxation of the max-cut problem, whose objective value provides an upper bound on the max-cut value, (b) the Goemans-Williamson heuristic to round the solution to the SDP relaxation into a feasible cut vector, that provides a lower bound on the max-cut value, and (c) a separation oracle that returns cutting planes to cut of the optimal solution to the SDP relaxation that is not in the maxcut polytope. Steps (a), (b), and (c) are repeated until the algorithm finds an optimal solution to the maxcut problem.

We have implemented the above cutting plane algorithm in MATLAB. Step (a) of the program uses SDPT3 - a primal dual interior point software for solving the SDP relaxations. Step (c) of the algorithm returns triangle inequalities specific to the max-cut problem as cutting planes. We report our computational results with the algorithm on randomly generated graphs, where the number of vertices and the density of the edges vary between 5 to 50 and 0.1 to 1.0, respectively.