Dissertation Proposal

Thiago Serra

Five Questions on Convexification and Postoptimality of Integer Programs

Tuesday, May 2, 2017
10:00 am
388 Posner Hall

In this thesis, we use concepts from lift-and-project and decision diagrams to explore issues concerning generation of cutting planes and enumeration of solutions of (mixed-)integer linear programs. Each chapter aims to answer a different question, which are the following:

- How can we find the deepest disjunctive cut in some direction by using lift-and-project? We introduce the Reverse Polar Cut Generating Linear Program (RP-CGLP), which evaluates disjunctive cuts according to their slack for a point p in the disjunctive hull when having the same violation with respect to the fractional solution x' being separated. We show that the resulting cut defines a supporting hyperplane of the disjunctive hull; in particular, this is the face exposed by the ray from p to x'. We also show the equivalence between RP-CGLP and other cut generating methods yielding the so-called target cuts, whereas RP-CGLP has the advantage that its feasible set is not affected by changing point p. Finally, we describe how to generate these cuts on split disjunctions directly from the simplex tableau of the problem.

- How can we check if a (strengthened) disjunctive cut is not dominated by Gomory (mixed-integer) cuts? We present computational experiments based on recent theoretical work by Balas and Kis (2016) to identify if a lift-and-project cut is regular or not. Regular cuts are those that can be derived as a Gomory cut from some LP basis of the problem relaxation. Our goal is to understand when lift-and-project cuts from non-split disjunctions are different and, if so, how much benefit do they bring.
This chapter is under development.

- How can we exploit dominance relations among branching nodes to encode near-optimal solutions? We revisit the Sound Decision Diagram (SndDD) proposed by Hadzic and Hooker (2007), which encodes a set of solutions within a gap from the optimal value along with feasible and infeasible solutions that are not near-optimal. We characterize the SndDD by showing that (i) there is no additional benefit from using these diagrams for Delta=0; (ii) one can find a minimum size SndDD by applying an operation that we denote sound-reduction a finite number of times; (iii) there may exist multiple minimum size SndDDs for a given solution set; and (iv) generalizing the SndDD to also encode infeasible solutions with better cost than the optimal yields a diagram that cannot be efficiently traversed to enumerate relevant solutions. In addition, we show that identifying nodes with equivalent subproblems despite different costs for the fixed variables while constructing these diagrams is equivalent to performing sound-reductions on the resulting diagram.

- How can we identify integer programming subproblems with equivalent formulations to avoid repetitive branching? We consider the parallel between exploring a branching tree and constructing a reduced decision diagram to enumerate a set of solutions, where the latter is smaller because isomorphic completions are merged. We revisit independent results from the optimization and the artificial intelligence literatures to directly construct a reduced decision diagram by anticipating which nodes can be merged for

a single inequality with integer coefficients and show that (i) a similar method can be applied in the case of fractional coefficients and additively separable functions on the left-hand side;
(ii) a different approach is need when there are multiple inequalities, and we show how to rule out equivalence of an unexplored node with all but most one explored node; and (iii) we can guarantee that the unexplored node yields the same solutions as the remaining explored node by generating a reasonable subset of bottom-up states of the decision diagram. We present some experiments where near-optimal solutions are enumerated by branching in fixed order, with the resulting method reducing branching by 40% and runtime by 20%. This chapter may be extended.

- How can we more efficiently manipulate a near-optimal solution set to perform sensitivity analyses? One can avoid resolving an integer program for small changes in the objective function if a sufficiently comprehensive set of near-optimal solutions is provided. While the size of such a set can be unmanageably large for direct inspection, in many cases it can be orders of magnitude smaller if represented as a decision diagram, where resolving the problem for an optimal solution corresponds to finding a shortest path. Our goal is to explore how much can we do with a decision diagram of near-optimal solutions, how fast can we do it, and which applications could benefit from that. This chapter is under development.