Orbital Crossover*

Ethan Deakins[1], James Ostrowski[2], and Ben Kneuven[3]

[1] University of Tennessee, USA
edeakins@utk.edu
[2] University of Knoxville Tennessee, USA
jostrows@utk.edu
[3] Sandia National Laboratories, USA
bknueve@sandia.gov

Abstract. **Keywords:** Linear Programming · Symmetry · Equitable Partitions

1 Introduction

Linear Programming (LP) has been long studied and is quite well understood. There are many different algorithms that solve instances of LP exactly, some of which are: the simplex algorithm (active set method) and the barrier algorithm (interior point method). In practice, LP instances typically solve very quickly using either method of subsets of the methods above (i.e., dual simplex). However, LP is used in all open-source and commercial Integer Programming (IP) solvers to create bounds and cuts for the IP instance. Thus, it is useful to study how to solve LP relaxations as quickly and efficiently as possible.

Now, LP instances may have what is known as symmetry in their feasible region. By symmetry, we mean given $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{F}$, two solutions to the LP instance, we have the following

$$\mathbf{c}^T\mathbf{x}_1 = \mathbf{c}^T\mathbf{x}_2 \tag{1}$$

where $\mathbf{c}^T$, $\mathcal{F}$ are the vector of objective coefficients and feasible region of the instance, respectively. We can use these symmetric solutions to our advantage to exploit the structure of the instances.

We may use symmetry groups and the mathematically weaker concept of equitable partitions(e.g., the method shown in [1]) to aggregate these problems and solve a dimension reduced version of the instance. These reduced problems are solvable in less time than the original model and retain the same objective value. Formally, given an LP instance LP^0 formulated as

$$\begin{aligned} &\min \mathbf{c}^T\mathbf{x} \\ &\text{s.t. } A\mathbf{x} = \mathbf{b} \\ &\quad \mathbf{x} \geq 0. \end{aligned} \tag{2}$$

* Supported by organization x.

we may solve its aggregated formulation, say LP^1,

$$\begin{aligned} \min\ & \bar{\mathbf{c}}^T \bar{\mathbf{x}} \\ \text{s.t. } & \bar{A}\bar{\mathbf{x}} = \bar{\mathbf{b}} \\ & \bar{\mathbf{x}} \geq 0. \end{aligned} \tag{3}$$

and given $\mathbf{x}^*, \bar{\mathbf{x}}^*$, optimal solutions to LP^0 and LP^1, respectively, we have the property in (1). Note that standard form will be considered in this paper for matter of convenience, but is not required.

While the previous information shows that we can solve dimension reduced versions of LP instances and retain the optimal objective value, it does not hold that a vertex in LP^1 is a vertex in LP^0. Actually, it is likely that the aggregated solution will be an interior point solution to LP^0. Vertex solutions are preferable to interior point solutions due to their sparsity which is useful in both applications of LP and when LP is used for solving Integer Programming instances (a standard for modern IP solvers). As such, we must now choose a method to obtain a vertex in the original model.

One such method that offers a bridge over this impasse is that of the barrier algorithm which computes an interior point solution and then pushes it to a vertex solution (crossover). However, according to [2], crossover is the most computationally expensive component of the barrier algorithm, consuming on average 29% of the time of the barrier algorithm of models whose runtime was at least one second. The next two closest in terms of computational expense were that of factorization (25%) and presolve (14%), with all other components having an expense that accounted for less than 10% of the barrier algorithm's runtime.

The following paper proposes a more efficient method of moving from an interior point solution that is obtained from aggregating the LP instances using the method in [1]. This is interesting in general for LP because an LP may have a non-trivial (i.e., a non-discrete) coarsest equitable partition without having any symmetry. In our method we *crush* (aggregate) a given model based off its initial equitable partition, solve the crushed model and then *un-crush* (disaggregate) the model by iteratively refining the coarsest equitable partition of the original instance based on an isolation step that will be covered later. While un-crushing the instance, we make use of the well-known simplex algorithm with some minor edits in the method in which we choose entering variables, as well as, maintaining our active set.

I am going to add a contributions paragraph here, but I am not totally sure what to say our contributions are other than an efficient interior point algorithm.

2 Equitable partitions and color refinement

2.1 Equitable partitions

From this point forward, by graph we mean a finite, undirected weighted graph. An equitable partition of a graph $G = (V, E)$ can be defined as follows. Let Π denote a partition of the vertices of a graph then we have

Definition 2.1 *A partition Π of the vertices of a graph $G = (V, E)$ is considered equitable if* $\forall\ \pi_i, \pi_j \in \Pi$

$$\sum_{u \in \pi_i} w_{uv} = b_{ij},\ \forall\ v \in \pi_j \tag{4}$$

Note that a *coarsest* equitable partition is an equitable partition that cannot be further refined. where w_{uv}, b_{ij} are the weights along the edge (u, v) and a constant specific to parts π_i, π_j, respectively. For those who have used equitable partitioning for unweighted graphs, note that this definition differs slightly in that we are summing the edge weights between nodes of two parts rather than the degree sums of nodes in two distinct parts.

An equitable partition of a graph is similar to a collection of orbits as the result of a symmetry group. The difference lies in the restriction on permutation matrices. A symmetry group may be thought of as a collection of permutation matrices on a group (vertices in this case) such that each row and column sum to 1 and the entries are binary. An equitable partition can be thought of as permutation matrices but with the notion that the binary constraint has been relaxed to a continuous value between 0 and 1.

2.2 Color refinement

[1] gives a method of computing equitable partitions for an LP instance via a color refinement scheme. First, note that any LP can be represented as a bipartite graph $G = (V, E),\ V = V_1 \cup V_2,\ V_1 \cap V_2 = \emptyset$. Here V_1, V_2 are two disjoint collections of the nodes of the bipartite graph, namely the variables and constraints respectively. Now we can use color refinement to compute an equitable partition of the vertices. First each variable node $u \in V_1$ receives a color that maps it to its objective coefficient. Then, these sames nodes are examined again and the color of the nodes is updated based upon upper and lower bounds of the variables. For example, if two variables have the same objective coefficient but differ in either of their bounds, then these two variables will need to be assigned different colors. Constraint nodes $v \in V_2$ are handled in a similar fashion, however these nodes only need be examined once initially and are assigned on a color that maps it to the right hand side of that constraint.

Now, with each node colored, we have a trivial partition of the bipartite graph representing this LP, however this partition may not be equitable. If not, we may achieve an equitable partition by choosing a part $\pi_i \in \Pi$ and use it to refine all other parts $\pi_j \in \Pi$. This refined procedures evaluates the condition

in (4) for all pairs $\pi_i, \pi_j \in \Pi$ where i is fixed. If a node in a part $\pi_j, j \neq i$ violates this condition, it is assigned a new color. Any other node that violates this condition is either assigned to the new color given to the first node that was recolored, or it is given a new color as well. This process continues until no new colors are given in the partition. For details see [1].

References

1. Grohe, M., Kersting, K., Mladenov, M., Selman, E.: Dimension reduction via colour refinement. In: European Symposium on Algorithms. pp. 505–516. Springer (2014)
2. Maes, C., Rothberg, E., Gu, Z., Bixby, R.: Initial basis selection for lp crossover. In: Presented at CERFACS 2014 Toulouse, France