Orbital Crossover

Ethan Deakins[1], Bernard Knueven[2], Jim Ostrowski[1]

[1] Industrial and Systems Engineering, University of Tennessee, TN 37996, USA
edeakins@vols.utk.edu, jostrows@utk.edu
[2] Computational Science Center, National Renewable Energy Laboratory, Golden, CO 80401, Bernard.Knueven@nrel.gov

Abstract. Symmetry in optimization has been known to wreak havoc in optimization algorithms. Often, some of the hardest instances are highly symmetric. This is not the case in linear programming, as symmetry allows one to reduce the size of the problem, possibly dramatically, while still maintaining the same optimal objective value. This is done by aggregating symmetric variables. However, this aggregation, when disaggregated, will necessarily be on the center of the optimal face. Similar to interior point methods, a crossover routine has to be used to convert that solution to a vertex solution. In this work, we use a generalization of symmetries, equitable partitions, to aggregate and solve a linear program. Then, using the partition as a guide, we show how to effectively convert the disaggretated solution as a vertex. This allows us to not just solve, but arrive at a vertex solution, to otherwise difficult highly symmetric problems.

Keywords: Symmetry, Linear Programming, Crossover

1 Introduction

We consider linear programs $LP(A, b, c)$ of the following form:

$$\max\ c^t x \tag{1a}$$
$$\text{s.t. } Ax \leq b \tag{1b}$$
$$x \geq 0, \tag{1c}$$

with $c^t \in \mathbb{R}^n$ representing the objective coefficients and x and in $\mathbb{R}^n$ representing the decision variables. We have that b is in $\mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$.

In general, we solve problems of the form in (1) using either simplex methods (primal or dual simplex algorithms) or interior point methods (IPM)s (logarithmic barrier, predictor-corrector, etc). An interior point method improves a feasible interior solution point of a linear program (LP) by steps through the interior of the feasible region in contrast to simplex methods that move through steps around the boundary [5]. Both theoretically and practically, IPMs are attractive due to their polynomial run-times. However, they produce interior point

solutions, whereas, simplex methods produce vertex solutions that are sparser and may be preferable in various applications.

Typically, a crossover routine accompanies an IPM to achieve a vertex solution to the LP problem [2]. Crossover techniques are computationally expensive. Gurobi reports that 29% of the solver time is spent in crossover when the solver uses an IPM [4]. However, IPMs are not the only means of producing interior point solutions in the hopes of solving LPs faster. Symmetric structures, and their generalizations, of LPs can be exploited to reduce problem size, but doing so may return interior point solutions.

In [7], the authors show how to reduce the dimension of an LP with respect to an equitable partition (EP). Further, they show that a feasible solution to the reduced LP can be lifted to feasible solution to the original LP with the same objective value, and vice versa. The conversion from a reduced solution to a full-dimension solution and vice versa occurs by matrix multiplication by a suitable matrix. However, this conversion does not guarantee one arrives at a basic solution.

We organize the remainder of this paper as follows. First, a brief discussion of preliminaries on symmetry and EPs. Next, an explanation of the ideas behind the construction of orbital crossover. After this, an extension of the foundation of orbital crossover and how to apply it in an iterative procedure corresponding to iterative refinements of an EP. Next, an explanation of the methodology behind benchmark setups and results. The paper rounds out with a discussion on the performance of orbital crossover.

2 Background: Symmetry and EPs

Partitions and Restrictions:

Let $\mathcal{P} = \{P_1,\ P_2,\ \ldots,\ P_k\}$ be a partition of the decision variables and let the *representative* of $P_k \in \mathcal{P}$ be the lowest index member of P_k. Let $\mathcal{R}(\mathcal{P})$ be the set of all representatives associated with a partition $\mathcal{P}$. Further, we let $R(x_i, \mathcal{P})$ be the function that identifies the representative of x_i given partition $\mathcal{P}$, that is, the representative $x_j \in \mathcal{R}(\mathcal{P})$ such that x_i and x_j are in the same set $P_k \in \mathcal{P}$. We overload the R notation to act on sets $P_k \in \mathcal{P}$ as well, where $R(P_k)$ returns the representative of the set P_k, that is $R(P_k) = P_k \cap \mathcal{R}(\mathcal{P})$.

We define the restricted linear program, $RLP(A, b, c, \mathcal{P})$ to be that as (1) with the additional constraint:

$$x_i = x_{R(x_i,\mathcal{P})}. \tag{2}$$

Obviously, $RLP(A, b, c, \mathcal{P})$ can be written in such a way as to aggregate variables that belong in the same set in the partition. Such an aggregation will yield a formulation with only k variables. It is possible that such a restriction may make a subset of the constraints linearly dependent. We assume that those constraints are handled via preprocessing. We will refer to the aggregated version as $ALP(A, b, c, \mathcal{P})$.

We will make use of the *extended linear program*, $ELP(A, b, c, \mathcal{P})$, where the set of constraints (2) are replaced by

$$r_{i,R(x_i,\mathcal{P})} = x_i - x_{R(x_i,\mathcal{P})} \tag{3}$$

and note that since the newly formed r variables are unbounded and appear in just one (new) equality each, a basis for $LP(A, b, c)$ can easily be mapped to a basis for $ELP(A, b, c, \mathcal{P})$ by adding the r variables to the basis. Similarly, a basis for $ELP(A, b, c, \mathcal{P})$ can be mapped to a basis in $LP(A, b, c)$ if all the r variables are basic, simply by truncating the r variables and their associated constraints. We will show how to efficiently arrive at an optimal basic solution of $LP(A, b, c)$ given an optimal basic feasible solution (BFS) to $ALP(A, b, c, \mathcal{P})$, by way of $ELP(A, b, c, \mathcal{P})$, when $\mathcal{P}$ is generated via symmetry information.

Symmetries and Fractional Symmetries:

In this section we mention some basic ideas in group theory. An in-depth review can be found in [3,8,15]. The symmetry group of $LP(A, b, c)$ is defined to be permutations that map feasible solutions to the LP to feasible solutions with the same objective value [13]. Let $\mathcal{G}$ represent the symmetry group. Computing $\mathcal{G}$ is not practical, as it requires the knowledge of all feasible solutions. The *formulation group*, $\mathcal{G}'$, is used to approximate $\mathcal{G}$ in practice. Note that $\mathcal{G}' \subset \mathcal{G}$. The formulation group of (1) contains the set of permutations P_π such that there exists a $(P_\pi,\ P_\sigma)$ with:

$$cP_\pi = c, \tag{4a}$$
$$P_\sigma b = b, \tag{4b}$$
$$AP_\pi = P_\sigma A, \tag{4c}$$
$$P_\pi \in \mathcal{S}^n,\ P_\sigma \in \mathcal{S}^m, \tag{4d}$$

where $\mathcal{S}^n$ and $\mathcal{S}^m$ are the sets of all permutation matrices of appropriate size [11].

This ensures that integer solutions are mapped to other integer solutions, something that is important in integer programming but not in linear programming. Note that P_π represents permutations of the columns (variables) while P_σ represents permutations of the rows (constraints). Thus, we can think of constraints being equivalent with respect to the group generated by all feasible P_σ permutations. Recall that we are considering linear programs whose constraints are written as $Ax \leq b$. The pair (P_π, P_σ) can be considered a symmetry of the problem written in standard form, where P_σ describes the action on the slack variables. There is a correspondence between constraints that are equivalent with respect to a group and slack variables that are equivalent with respect to that same group.

Dropping the integrality restriction on (P_π, P_σ), gives the set of *fractional symmetries*, $\mathcal{F}$, that contain the set of doubly stochastic matrices that satisfy (4) except that we have $P_\pi \in \mathcal{D}^n$ and $P_\sigma \in \mathcal{D}^m$, where $\mathcal{D}^n$ and $\mathcal{D}^m$ are the sets of all doubly stochastic matrices of appropriate size [7]. Note that the set of all

fractional symmetries can be thought of as a polyhedron (though they no longer form a group).

Orbits and Generalizations:

A problem's symmetry group can be used to define equivalence classes on both the sets of variables as well as the set of feasible solutions. Two solutions are equivalent with respect to the group $\mathcal{G}$ if there exists a permutation in $\mathcal{G}$ that maps one solution to the other. We call the set of all solutions equivalent with respect to the group $\mathcal{G}$ to a solution x the *orbit* of x with respect to $\mathcal{G}$ [10]. This is denoted as $orb(x,\ \mathcal{G})$. We overload the *orb* notation to act on variables as well as vectors. We have that $j \in orb(i, \mathcal{G})$ if and only if $e_j \in orb(e_i, \mathcal{G})$. We say that two variables in the same orbit are equivalent. The variable orbits can be used to define a natural partition of the variables as $j \in orb(i, \mathcal{G})$ implies that $i \in orb(j, \mathcal{G})$. We let $\mathcal{O} = \{O_1, \ldots,\ O_k\}$ represent the *orbital partition* of the variables.

While not immediately obvious, *equitable partitions* (EP)s are the fractional symmetry analog of orbits. Formally, $(\mathcal{V}, \mathcal{C})$, partitions of the variables ($\mathcal{V} = (V_1,\ \ldots,\ V_k)$) and constraints ($\mathcal{C} = (C_1,\ \ldots,\ C_k)$), is an EP with respect to $\mathcal{F}$, if it satisfies the following constraints:

$$\sum_{p \in C} (A_{i,p} - A_{j,p}) = 0\ \forall i, j \in V \in \mathcal{V},\ C \in \mathcal{C} \tag{5a}$$

$$\sum_{i \in V} (A_{i,p} - A_{i,q}) = 0\ \forall p, q \in C \in \mathcal{C}, V \in \mathcal{V} \tag{5b}$$

$$\sum_{i,j \in C} (b_i - b_j) = 0\ \forall C \in \mathcal{C} \tag{5c}$$

$$\sum_{i,j \in V} (c_i - c_j) = 0\ \forall V \in \mathcal{V} \tag{5d}$$

We say that partition $\mathcal{P}^2$ is a *refinement* of partition $\mathcal{P}^1$ if for all $P_i^2 \in \mathcal{P}^2$ there exists a $P_j^1 \in \mathcal{P}^1$ with $P_i^2 \subseteq P_j^1$. It can be shown that any orbital partition is a refinement of an EP. Conversely, $\mathcal{P}^1$ is *coarser* than $\mathcal{P}^2$. The coarsest EP can be computed efficiently [1], and is often used as the first step in computing the formulation group of an instance.

Stabilizers and Generalizations:

The *stabilizer* of a group $\mathcal{G}$ with respect to an element v, $stab(v, \mathcal{G})$ is defined to be:

$$stab(v, \mathcal{G}) := \{P_\pi \in \mathcal{G} \mid P_\pi(v) = v\}.$$

Note that this definition allows for v to be either a vector or a variable. Using the constraints (4), this can be thought of adding the constraint $P_\pi v = v$, or, in the case where v represents a variable, say x_i, by fixing $P_\pi(i, i)$ to one.

Similar to the stabilizer, we wish to *isolate* elements of an EP by creating a refinement where that element is in a singleton set in the refined partition. Formally, we have $iso(v, \mathcal{F})$ defined as:

$$iso(v, \mathcal{F}) := \{P_\pi \in \mathcal{F} \mid P_\pi(v) = v\}.$$

Again, this can be thought of adding the constraint $P_\pi v = v$, or, in the case where v represents a variable, say x_i, by fixing $P_\pi(i, i)$ to one.

EPs are useful in LP as they provide a method of aggregating variables based on symmetry information. Further, when an LP is aggregated based on $\mathcal{P} = EQ(\mathcal{F})$, we have the following theorem.

Theorem 1 (Lemma 5.1 [7]). *Let $\mathcal{F}$ be the set of fractional symmetries acting on a linear program, $\mathcal{P} = (\mathcal{V}, \mathcal{C})$ an EP with respect to $\mathcal{F}$, z_R^* the optimal objective value to $RLP(A, b, c, \mathcal{P})$ and z^* the optimal objective value to $LP(A, b, c)$. Then, $z_R^* = z^*$.*

3 Orbital Crossover

For $\mathcal{P} = (\mathcal{V}, \mathcal{C})$, an EP of $LP(A, b, c)$, we will show how to use an optimal basic solution to $ALP(A, b, c, \mathcal{P})$ to find an optimal basic solution to $LP(A, b, c)$ via $ELP(A, b, c, \mathcal{P})$.

Creating $ALP(A, b, c, \mathcal{P})$:

First, we need to define how the aggregate formulation is created. Let $\mathcal{R}(\mathcal{V})$ be the set of variable representatives and $\mathcal{R}(\mathcal{C})$ be the set of constraint representatives in $\mathcal{R}(\mathcal{P})$. Then $ALP(A, b, c, \mathcal{P})$ has $|\mathcal{R}(\mathcal{V})|$-many variables and $|\mathcal{R}(\mathcal{C})|$-many constraints. For each representative constraint $c_j \in \mathcal{R}(\mathcal{C})$, we construct an aggregated constraint by simply combining the coefficients of variables in the same set within the EP $\mathcal{P}$. That is, the coefficient of each representative variable $x_i, i \in R(V)$ in $ALP(A, b, c, \mathcal{P})$ will be

$$\sum_{l|R(l,\mathcal{P})=i} A_{j,l},$$

and the right hand side of the constraint remains unchanged. The objective function is similarly aggregated.

Example 1. Aggregating an instance of $LP(A, b, c)$

Consider the following example LP:

$$\begin{aligned}
\max\ & x_1 + x_2 + x_3 && && \text{(6a)}\\
\text{s.t. } & x_1 + x_2 \le 1 && (c_1) && \text{(6b)}\\
& x_2 + x_3 \le 1 && (c_2) && \text{(6c)}\\
& x_1 + x_3 \le 1 && (c_3) && \text{(6d)}\\
& x_1 + x_2 + x_3 \le 1 && (c_4) && \text{(6e)}\\
& x_i \in \mathbb{R}^+. && && \text{(6f)}
\end{aligned}$$

We add slack variables for each constraint, denoted as s_i, to transform the problem into standard form. Here, s_i is the slack for constraint c_i. In this LP, the coarsest EP

$$\mathcal{P} = (\mathcal{V}, \mathcal{C}) = \{(x_1, x_2, x_3), (s_1, s_2, s_3), (s_4), (c_1, c_2, c_3), (c_4)\}.$$

Note that the partitions of the slack variables correspond to the partitions of the constraints. Given this correspondence, we will omit constraint partitions going forward. The representatives are x_1, s_1, s_4, c_1, and c_4. Thus, we can write $ALP(A, b, c, \mathcal{P})$ for this example in standard form as an LP on two variables with one constraint:

$$\begin{aligned} \max\ & 3x_1 && && (7a)\\ \text{s.t. } & 2x_1 + s_1 = 1 && (c_1) && (7b)\\ & 3x_1 + s_4 = 1 && (c_4) && (7c)\\ & x_1, s_1, s_4 \in \mathbb{R}^+. && && (7d) \end{aligned}$$

We note that the optimal solution to (7) is $(x_1, s_1, s_2) = \left(\frac{1}{3}, \frac{1}{3}, 0\right)$. This can be interpreted as saying that the average of all variables in $V_1 \in \mathcal{P}$ is $\frac{1}{3}$, the average of all variables in $V_2 \in \mathcal{P}$ is $\frac{1}{3}$, and the average of variables in $V_3 \in \mathcal{P}$ is 0. Also, in this optimal solution we have x_1 and s_1 as basic variables and s_4 as a nonbasic variable.

Lifting Solutions from $ALP(A, b, c, \mathcal{P})$ to $ELP(A, b, c, \mathcal{P})$:

After arriving at an optimal BFS to the $ALP(A, b, c, \mathcal{P})$, we then lift the solution to an optimal BFS to the extended linear program $ELP(A, b, c, \mathcal{P})$. Recall that the ELP contains all of the original variables in addition to the linking r variables (3).

Let x_i^{a*} be the value of the aggregate variable x_i^a. The original variables can be recovered by:

$$x_i = x^a_{R(x_i, \mathcal{P})}\ \forall i \in \{1,\ \ldots,\ n\}, \tag{8}$$

recalling that $R(x_i, \mathcal{P})$ identifies the representative of i in partition $\mathcal{P}$. Note that this implies that

$$r_{i, R(x_i, \mathcal{P})} = 0\ \ \forall\ i \neq R(i, \mathcal{P}). \tag{9}$$

Similarly, for s_i^{a*} the slack variables can be recovered by

$$s_i = s^a_{R(s_i, \mathcal{P})}\ \forall i \in \{1,\ \ldots,\ m\}. \tag{10}$$

for optimal aggregate slack variable s_i^{a*}.

Let $(x, s, r)^*$ denote the lifted solution from $(x^a, s^a)^*$, an optimal BFS from the $ALP(A, b, c, \mathcal{P})$. We have the following result.

Theorem 2. *$(x, s, r)^*$ is an optimal BFS to $ELP(A, b, c, \mathcal{P})$ with nonnegativity constraints on the r variables.*

Proof. First, note that as a consequence of [7], if we truncate $(x, s, r)^*$ to $(x, s)^*$ then $(x, s)^*$ is both optimal and feasible for $LP(A, b, c)$. Now, recall that equation (8) implies equation (9), thus $(x, s, r)^*$ satisfies $ELP(A, b, c, \mathcal{P})$. Next, note that there are $n + m + (n - |\mathcal{R}(V)|)$-many variables in $ELP(A, b, c, \mathcal{P})$. The constraints of $ELP(A, b, c, \mathcal{P})$ contribute $m + (n - |\mathcal{R}(V)|)$-many linearly independent binding constraints. Thus the set of nonbasic variables should be of size n.

Let NB_{ALP} be the set of nonbasic variables in the optimal BFS to $ALP(A, b, c, \mathcal{P})$. We have that $|NB_{ALP}| = |R(\mathcal{V})|$. The disaggregation of every nonbasic variable in NB_{ALP} contributes one non-basic variable in $ELP(A, b, b, \mathcal{P})$. In addition, there are $n - |R(\mathcal{V})|$-many r variables that are in the non-basis, giving a total of $|R(\mathcal{V})| + n - |R(\mathcal{V})| = n$-many variables.

□

Example 2. Lifting an instance of $ALP(A, b, c, \mathcal{P})$ to $ELP(A, b, c, \mathcal{P})$.

The extended formulation for the LP in the previous example is as follows:

$$\begin{aligned}
&\max\ x_1 + x_2 + x_3 && && \text{(11a)}\\
&\text{s.t. } x_1 + x_2 + s_1 = 1 && (c_1) && \text{(11b)}\\
&\quad x_2 + x_3 + s_2 = 1 && (c_2) && \text{(11c)}\\
&\quad x_1 + x_3 + s_3 = 1 && (c_3) && \text{(11d)}\\
&\quad x_1 + x_2 + x_3 + s_4 = 1 && (c_4) && \text{(11e)}\\
&\quad r_{2,1} = x_2 - x_1 && && \text{(11f)}\\
&\quad r_{3,1} = x_3 - x_1 && && \text{(11g)}\\
&\quad x, s, r \geq 0. && && \text{(11h)}
\end{aligned}$$

Performing the lifting via equations (8) and (10), we have $x_1 = x_2 = x_3 = \frac{1}{3}$, $s_1 = s_2 = s_3 = \frac{1}{3}$, $s_4 = 0$ $r_{2,1} = 0$, and $r_{3,1} = 0$. There are nine variables and six equality constraints. Letting s_4, $r_{2,1}$, and $r_{3,1}$ be the set of nonbasic variables will result in a degenerate BFS.

Arriving at an optimal BFS to $LP(A, b, c)$:

The previous section shows how to construct an optimal BFS to $ELP(A, b, c, \mathcal{P})$ from the solution to $ALP(A, b, c, \mathcal{P})$. However, we wish to have an optimal BFS to $LP(A, b, c)$. We note that this can be accomplished easily if all the r variables are in the basis by truncating the r variables. Starting at the lifted BFS, where all the r variables are nonbasic, we iteratively pivot on each r variable. To ensure that the r variables never re-enter the set of nonbasic variables, we relax their lower bounds to be negative infinity as they enter the basis, making them free variables. We need to ensure, however, that the objective value does not change as we pivot r variables into the basis.

Theorem 3. *One can pivot on any r variable without changing the objective function.*

Proof. Let z^* be the optimal solution value to $ALP(A, b, c, \mathcal{P})$. From Theorem 1, we have that the optimal solution value to $LP(A, b, c)$ is z^* and that the lifted basic solution also has an objective of z^*. Consider pivoting on the nonbasic variable r. If either the reduced cost of r is zero or the pivot is degenerate, then pivoting does not change the objective. Consider now the case where the reduced cost is strictly positive and the pivot is nondegenerate. Note that the non-negetivity constraint on r is arbitrary and could have been written as a non-positivity constraint, and the current BFS would still remain a BFS if such a change were made. Switching the sense of r would negate the reduced cost of r. Since the current BFS is optimal, pivoting on r in the "down direction" must be degenerate, so the solution value does not change.

Reducing the number of r variables:

In the above method, $n - |\mathcal{R}(V)|$-many r variables are created, and each of these r variables adds one simplex pivot to the crossover algorithm. The quantity of r variables can be reduced by taking advantage of the following observation. If a constraint (including a variable bound) is binding in the solution to $ALP(A, b, c, \mathcal{P})$, then there exists a vertex in $LP(A, b, c)$ where all equivalent constraints are binding. The intuition behind this is that in $ALP(A, b, c, \mathcal{P})$, x and s represent the average value of the variables they represent, so if one is at its bound, then all are at their bound. Consider then an aggregate x variable that is nonbasic in $ALP(A, b, c, \mathcal{P})$. In the lifted tableau, variables that are not representatives variable are always basic. This leads to a situation where the tableau contains constraints of the form:

$$x_i - x_{R(x_i,\mathcal{P})} - r_{i,R(x_i,\mathcal{P})} = 0,$$

where $r_{i,R(x_i,\mathcal{P})}$ is nonbasic and $x_{R(x_i,\mathcal{P})} = 0$. Clearly, we can perform a degenerate (downward) pivot on $r_{i,R(x_i,\mathcal{P})}$ to remove it from the nonbasis and add x_i to the nonbasis. Thus, we could have originally omitted the $r_{i,R(x_i,\mathcal{P})}$ variable (and its associated constraint) from the formulation and simply included x_i in the nonbasis.

Note that slack variables do not have associated r variables, so we cannot easily swap basic s variables that we know to be zero into the nonbasis in place of r variables. It is likely that enforcing some constraints to be binding will force a set of r variables to take the value of zero, so it might be possible to construct the lifted BFS with fewer r variables in the nonbasis. However, identifying which r variables to remove from the basis would require testing for linear dependence. This linear dependence check might be more expensive than the savings gained by fewer pivots.

4 Iterative Lifting

The benefit of the method described in Section 3 is that an optimal solution to the linear program can be found in the aggregated space. Crossover itself,

however, is done in the extended space. By performing an iterative lifting procedure, we can do many of the crossover operations in lower dimensions. Recall that for *any* EP $\mathcal{P}$, $RLP(A, b, c, \mathcal{P})$ will have the same optimal solution value as $LP(A, b, c)$. Obviously, we'd prefer to find the coarsest partition to allow for the most aggregation (and smallest dimension of $ALP(A, b, c, \mathcal{P})$). Let $\mathcal{P}^1$ and $\mathcal{P}^2$ be two EPs of $LP(A, b, c)$ where $\mathcal{P}^2$ is a refinement of $\mathcal{P}^1$. We can use the above methods to lift a vertex solution of $ALP(A, b, c, \mathcal{P}^1)$ to $ALP(A, b, c, \mathcal{P}^2)$ and as long as $\mathcal{P}^2$ is not the discrete partition (i.e., $|P_k| = 1, \forall P_k \in \mathcal{P}^2$), this lifting will be done in a smaller dimension than the original extended formulation.

We generate our set of EPs as follows. Let $\mathcal{P}$ be the the coarsest possible EP with respect to $\mathcal{F}$. We let $\mathcal{P}^1$ be the EP generated by $\mathcal{F}^1 = iso(x_1, \mathcal{F})$. Similarly, we let $\mathcal{P}^i$ be the coarsest EP generated by $\mathcal{F}^i = iso(x_i, \mathcal{F}^{i-1})$.

5 Benchmarks

Implementation:

Orbital crossover was implemented using HiGHS version 1.2. HiGHS is a high performance serial and parallel dual solver for large scale sparse linear programming problems (see [9] and https://github.com/ERGO-Code/HiGHS for more details). Although the HiGHS authors market it as a high-performance dual simplex solver, it possesses a well-implemented version of the primal simplex algorithm and an IPM solver with crossover. This makes HiGHS a capable candidate for implementing OC and for comparing against the dual simplex and IPM solvers.

A modified version of **saucy** [6,16] was used to compute the EPs. **Saucy** was chosen over other symmetry-detection software [12,14], as it has an extension [16] that allows it to accept LP files. It was integrated directly into the HiGHS source files as a class object callable by the main HiGHS objects.

Test Sets, Methods, and setup:

Benchmark tests are set up using two different LP data sets. The first data set is a group of LP relaxations of instances in **MIPLIB 2017**. All instances have any integrality constraints relaxed. The second data set is a large collection of LP relaxations of highly symmetric MILP instances. The large majority of these problems are coding binary ternary instances and covering problem instances. The coarsest equitable partitions of these instances are very large, meaning that the resulting aggregated problems are small. We refer to this data set as **HS-COV-COD**.

We use five different solving strategies across both data sets above. We list these strategies below.

- **HDS:** HiGHS serial dual simplex solver,
- **HIP:** HiGHS IPM solver with HiGHS crossover,
- **HIP ALP:** HiGHS IPM solver to solve $ALP(A, b, c, \mathcal{P})$, lift the solution to $LP(A, b, c)$, and obtain a optimal BFS with HiGHS crossover,
- **OCDS:** OC with iterative lifting using HiGHS serial dual simplex solver to solve $ALP(A, b, c, \mathcal{P})$,

– **OCIP:** OC with iterative lifting using HiGHS IPM solver and HiGHS crossover to solve $ALP(A, b, c, \mathcal{P})$.

Results:

All results for the **HS-COV-COD** test suite are in Table 1. The difference between OCDS and OCIP is minimal in the highly symmetric instances, as they likely are lifting the same solution to the the aggregate problem. Interestingly, the times reported for HIP, HIP ALP, OCDS, and OCIP are almost entirely the time spent in crossover. The small size of the aggregate allows HIP ALP, OCDS, and OCIP to solve the aggregate problem instantaneously, while HIP spends all but 2 seconds (for the entire test suite) in crossover.

When looking at **MIPLIB 2017**, we remove the HDS column from Table 2 as it is widely outperformed. However, these problems have more significant aggregated problems, and we wish to separate the impact of the solving the aggregate versus time spent specifically in crossover. We've added four columns titled HC, HAC, OC (DS), and OC (IP) which describe the time spent in crossover for HiGHS crossover from HIP, HiGHS crossover from HIP ALP, OC from OCDS, and OC from OCIP, respectively. We perform the same pair-wise comparison for these instances. HIP is the fastest solver for two instances. HIP ALP is the fastest solver for ten instances. OCDS is the fastest solver for six instances. OCIP is the fastest solver one instance.

Note that HC performs much better than in the highly symmetric instances. This is likely due to the fact that the lifted solution to the aggregated problem sometimes is a vertex in original (or only a few pivots away). The OC methods would still perform degenerate pivots if the lifted solution was a vertex.

Most modern commercial solvers use multiple methods to solve a given LP instance by passing each method to a single core in a multi-core machine and taking the solution from the solver that returns first. We simulate this method by taking the minimum solve times across each instance and analyze the total solve time. For the instances in **HS-COV-COD**, the sum of the minimum solve times is 1,230.46 seconds. For the instances in **MIPLIB 2017**, the sum of the minimum solve times is 274.42 seconds. This is a 8.74% improvement over OCDS for the **HS-COV-COD** data set and a 33.23% improvement over OCIP for the **MIPLIB 2017** data set.

6 Discussion

OC proves to be an effective method of exploiting symmetry. For the instances in **HS-COV-COD**, we show that the OCDS and OCIP strategies are approximately 87% faster than the next best strategy not using OC. The reason for this is the instances in this data set have a immense amount of symmetry which allows OC to further exploit the structure of the problems. This allows OC to perform much of the pivoting work in smaller dimensions via iterative lifting.

The benchmark results for the instances in **MIPLIB 2017** show different results. These instances do have symmetry, but in general, not nearly as much

Table 1. Benchmark Results for the **HS-COV-COD** Instances

Instance	HD	HIP	HIP ALP	OCDS	OCIP
cod733	30.35	1.95	2.12	1.74	**1.73**
codbt161	22.88	**0.75**	1.07	0.84	0.84
codbt162	37.65	**0.90**	2.41	3.27	3.21
codbt163	77.56	**2.01**	5.00	4.78	4.80
codbt164	82.17	**2.90**	6.37	3.64	3.60
codbt171	3601.87	**5.42**	14.92	24.63	24.33
codbt172	3611.58	**188.32**	288.13	216.73	212.74
codbt173	3607.37	**61.98**	194.39	100.84	80.92
codbt174	3615.73	**94.84**	193.55	116.49	95.15
codbt261	464.03	**3.11**	8.01	6.51	6.56
codbt262	2601.94	65.62	65.60	**48.92**	49.35
codbt451	3603.75	**15.97**	30.89	27.47	26.98
codbt452	3626.21	96.47	156.73	90.38	**88.94**
codbt453	3620.45	**40.11**	81.68	47.67	47.96
codbt731	411.99	**8.84**	14.62	16.30	16.13
codbt732	2304.69	**12.71**	23.91	21.08	21.13
codbt821	363.48	**0.76**	2.27	3.67	3.76
codbt822	236.31	**2.94**	6.40	5.40	5.32
codbt831	3626.37	154.34	199.01	95.31	**94.89**
codbt832	3645.83	979.85	1524.01	212.35	**210.42**
cov1385	12.42	**0.23**	0.58	0.72	0.72
cov1496	114.54	4.72	11.30	**3.79**	3.88
cov1497	49.66	3.17	6.48	**2.29**	2.43
cov15107	967.88	322.61	275.93	**19.13**	19.16
cov15108	282.21	99.24	90.47	**9.98**	10.00
cov15109	28.46	1.96	3.05	**0.80**	0.91
cov15118	13.37	28.44	32.83	1.31	**1.29**
cov161110	114.61	7.33	9.53	**1.81**	1.93
cov16118	3604.33	1956.83	2041.95	53.52	**53.24**
cov16119	1451.97	477.84	482.37	**15.01**	15.05
cov161210	23.78	21.43	22.73	1.19	**1.17**
cov16129	40.89	109.67	122.78	**3.75**	3.85
cov171210	3632.91	2208.61	2164.76	**38.00**	38.18
cov171211	444.56	24.37	31.18	2.41	**2.31**
cov17129	3632.81	3602.68	3659.56	**179.51**	182.04
cov171310	151.74	443.44	392.68	10.55	**10.52**
cov171311	129.75	65.65	70.72	3.02	**2.93**
	53888.09	11118.00	12239.96	1394.81	**1348.36**

Table 2. Benchmark Results for the **MIPLIB 2017** Instances

	Solve Time (Secs)				Crossover Time (Secs)			
Instance	HIP	HIP ALP	OCDS	OCIP	HC	HAC	OC (DS)	OC (IP)
atlanta-ip	12.80	12.14	**6.65**	12.50	**0.09**	0.13	0.30	0.34
bab6	27.47	19.19	**9.05**	19.12	**0.07**	0.23	0.23	0.23
chromatic-index1024-7	14.21	**2.05**	5.67	6.36	3.08	**1.31**	5.52	5.61
cryptanalysis-kb128n5obj14	5.58	**4.51**	16.11	5.16	0.79	0.55	**0.47**	0.52
cryptanalysis-kb128n5obj16	5.70	**4.46**	11.66	5.04	0.79	0.71	**0.47**	0.52
k1mushroom	34.34	22.96	**13.03**	21.37	**1.53**	5.22	2.81	2.94
map10	166.74	36.34	**17.20**	46.97	**0.31**	0.69	10.43	11.37
map16715-04	213.93	49.65	**19.10**	59.78	**0.29**	0.65	10.69	11.80
neos-3555904-turama	12.69	10.08	**6.18**	8.55	0.26	2.03	**0.20**	0.43
neos-4722843-widden	10.40	**0.72**	2.13	3.57	**0.07**	0.20	2.08	3.06
neos-631710	354.08	175.57	18.36	**14.33**	132.22	175.27	16.44	**13.87**
neos-873061	13.57	**11.92**	31.16	12.35	**0.17**	0.34	0.84	0.76
physician-sched6-2	62.11	**34.53**	80.97	38.67	2.65	**0.69**	2.76	3.95
rail01	36.16	**28.59**	85.01	33.57	7.50	1.34	**0.68**	0.75
rail02	93.16	**75.52**	972.41	94.00	13.19	**2.66**	6.26	8.84
satellites2-40	9.95	**9.18**	44.91	10.10	2.67	1.98	**0.09**	0.23
satellites2-60-fs	**6.22**	6.23	30.35	6.69	2.09	1.53	**0.13**	0.23
uccase12	14.84	**3.82**	8.48	4.84	**0.17**	0.32	1.43	1.31
uccase9	**7.36**	7.94	13.57	8.04	0.10	0.13	**0.08**	0.10
	1101.27	515.40	1391.98	**410.99**	168.03	195.99	**61.89**	66.85

symmetry as the instances in **HS-COV-COD**. We see for this data set that the OCIP strategy is the best performer in terms of time required to solve all instances by 20.26%.

The decrease in effectiveness for OC in the **MIPLIB 2017** data set can be partly explained by the instances having less symmetric structure. On the other hand, the standard method of determining superbasic variables to perform pushes on during HiGHS crossover may result in less pivots than what OC would require. We see that the addition of OCDS and OCIP strategies in the porfolio of solvers results in reduction of total solve time for both **HS-COV-COD** and **MIPLIB 2017** data sets making OC a valuable strategy.

Going forward, we see this as ba tool that can be used to help solve highly-symmetric integer programs. A trouble when using cut-generation schemes to solve highly-symmetric integer programs is that including all symmetric-equivalent cuts can quickly increase the size of the formulation. The proposed approach will help solve these large instances.

Acknowledgements

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided in part by the U.S. Department of Energy Advanced Research Projects Agency - Energy. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. A portion of this research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

References

1. Christoph Berkholz, Paul Bonsma, and Martin Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory of Computing Systems*, 60(4):581–614, 2017.
2. Robert E. Bixby and Matthew J. Saltzman. Recovering an optimal lp basis from an interior point solution. *Operations Research Letters*, 15(4):169–178, 1994.
3. Peter J Cameron et al. *Permutation groups*. Number 45. Cambridge University Press, 1999.
4. Christopher Maes, Edward Rothberg, Zonghao Gu, Robert Bixby. Initial basis selection for lp crossover. URL: https://cerfacs.fr/wp-content/uploads/2016/01/maes.pdf, 6 2014.
5. George B Dantzig and Mukund N Thapa. *Linear programming 1: introduction*. Springer Science & Business Media, 2006.
6. Paul T. Darga, Mark H. Liffiton, Karem A. Sakallah, and Igor L. Markov. Exploiting structure in symmetry detection for cnf. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, page 530–534, New York, NY, USA, 2004. Association for Computing Machinery.
7. Martin Grohe, Kristian Kersting, Martin Mladenov, and Erkal Selman. Dimension reduction via colour refinement. In *European Symposium on Algorithms*, pages 505–516. Springer, 2014.
8. Larry C Grove and Clark T Benson. *Finite reflection groups*, volume 99. Springer Science & Business Media, 1996.
9. Qi Huangfu and JA Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.
10. François Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming*, 98(1):3–21, 2003.
11. François Margot. *Symmetry in Integer Linear Programming*, pages 647–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
12. Brendan D McKay. Nauty user's guide (version 2.4). *Computer Science Dept., Australian National University*, pages 225–239, 2007.
13. James Ostrowski. *Symmetry in integer programming*. Lehigh University, 2009.

14. Peter Dobsan. Pynauty pypi web page. URL: https://pypi.org/project/pynauty/, 2022.
15. Joseph J Rotman. *An introduction to the theory of groups*, volume 148. Springer Science & Business Media, 2012.
16. Jonathan David Schrock. Symmetry detection in integer linear programs. 2015.