Math. Prog. Comp. (2014) 6:405–428
DOI 10.1007/s12532-014-0072-0

FULL LENGTH PAPER

Using symmetry to optimize over the Sherali–Adams relaxation

James Ostrowski

Received: 1 February 2012 / Accepted: 14 April 2014 / Published online: 4 May 2014

Abstract In this paper we examine the impact of using the Sherali–Adams procedure on highly symmetric integer programming problems. Linear relaxations of the extended formulations generated by Sherali–Adams can be very large, containing $O\left(\binom{n}{d}\right)$ many variables for the level-d closure. When large amounts of symmetry are present in the problem instance however, the symmetry can be used to generate a much smaller linear program that has an identical objective value. We demonstrate this by computing the bound associated with the level 1, 2, and 3 relaxations of several highly symmetric binary integer programming problems. We also present a class of constraints, called *counting constraints*, that further improves the bound, and in some cases provides a tight formulation. A major advantage of the Sherali–Adams formulation over the traditional formulation is that symmetry-breaking constraints can be more efficiently implemented. We show that using the Sherali–Adams formulation in conjunction with the symmetry breaking tool isomorphism pruning can lead to the pruning of more nodes early on in the branch-and-bound process.

Keywords Integer programming · Extended formulations · Symmetry · Branch and bound

Mathematics Subject Classification 90C10

1 Introduction

This paper looks to solve the binary integer programming problem

$$\min_{x\in\{0,1\}^n} \{c^T x \mid Ax \geq b\}, \tag{BIP}$$

where $c \in \mathbb{Q}^n$, $b \in \mathbb{Q}^m$, and $A \in \mathbb{Q}^{m\times n}$.

J. Ostrowski (✉)
Department of Industrial and Systems Engineering, University of Tennessee, Knoxville, USA
e-mail: jostrows@utk.edu

We assume that BIP has a very large symmetry group. The presence of symmetry can confound the branch-and-bound process since there can be many thousands of subproblems in the branch-and-bound tree that are equivalent, making even the smallest problems impossible to solve. In the past several years, however, algorithms such as isomorphism pruning [22], orbital branching [28], and orbitopal fixing [17] have been developed which successfully identify and remove equivalent subproblems from the branch-and-bound tree. Other methods include adding symmetry-breaking constraints [31] as well as perturbing the objective function [14]. These symmetry-exploiting algorithms can solve highly-symmetric problems orders of magnitude faster than methods that do not exploit symmetry. Despite these successes, there are still many highly-symmetric small BIPs that are unsolvable or require enormous amounts of computational time [19,29].

One possible explanation for the inability to solve modestly sized symmetric problems lies with the bounds. Cutting plane methods have been one of the driving forces behind the improvements in integer programming software. However, there is reason to believe that cutting plane methods may not be effective for highly symmetric problems. The linear program (LP) relaxation of symmetric BIPs may contain a large number of equivalent optimal basic solutions. Cutting *all* such solutions may require the generation of a large number of constraints, making the LP harder to solve. As an example of the failure of cutting plane methods, the Steiner triple systems (`sts`), the most symmetric class of problems in the MIPLIB library, is discussed in Sect. 3.

Rather than focus on cutting plane methods, this paper utilizes the Sherali–Adams [30] hierarchy to compute lower bounds and to prune nodes in the branch-and-bound tree. Bounds obtained by Sherali–Adams formulations can be very strong. It is known that the level-n formulation is the convex hull of the solution space. The level-1 closure has been shown to close a significant amount of the integrality gap [5]. Adams et al. [1] shows, in the case of the quadratic assignment problem (QAP), that the level-2 formulation can be much stronger. Bounds based on level-3 formulations of QAP problems are approximated in [15]. Extended formulations of Sherali–Adams relaxations can be extremely large and impossible to solve directly, as the extended formulation for a level-k relaxation will contain more than $\binom{n}{k}$ variables. Fortunately though, the problem symmetry can be exploited to help solve the relaxations faster. This issue has been addressed in the context of semidefinite programming (SDP) [7–9,32], where symmetry is used to create a much smaller SDP with an identical objective value. Similar methods to those used to create the smaller SDP can be applied to the Sherali–Adams formulation in order to create an LP that is solvable. While an SDP formulation is guaranteed to be at least as tight as the level-1 Sherali–Adams formulation, higher level Sherali–Adams formulations can be created. Interestingly, the power of the higher level Sherali–Adams formulation is not just in the improved bounds, but in its ability to efficiently enforce a class of symmetry breaking constraints. As a result, the Sherali–Adams formulation can be incorporated into symmetry breaking algorithms to help prune nodes in the branch-and-bound tree. This is something that is not done by using the traditional SDP formulation.

The paper is structured as follows. Section 2 gives a brief introduction to symmetry in integer programming and how it is exploited by techniques such as isomorphism pruning and orbital branching, while Sect. 3 describes why applying traditional cutting

plane algorithms to highly symmetric problems tend to fail. Section 4 details how to construct an LP formulation whose optimal objective value is equal to that of the Sherali–Adams formulation, and computes the Sherali–Adams bounds for a class of highly symmetric problems. These extended formulations are used in Sect. 5 to improve the bounds for certain subproblems in the branch-and-bound tree, and, as a result, help prune additional nodes.

2 A review of symmetry in integer programming

Describing symmetry in integer programs requires some definitions from algebra. The set Π^n is the set of all permutations of $I^n = \{1, \ldots, n\}$. This set forms the *full symmetric group* of I^n. Any subgroup, Γ, of the full symmetric group is a *permutation group*.

The permutation $\pi \in \Pi^n$ maps the point $z \in \mathbb{R}^n$ to $\pi(z)$ by permuting the indices. We extend the action of a permutation to sets, where, for $S \subseteq \mathcal{Z}$ and $S' \subseteq \mathcal{Z}$, $\pi(S) = S'$ if and only if $\pi\left(\sum_{i \in S} e_i\right) = \sum_{i \in S'} e_i$. The (following) definition of orbits is extended to sets in a similar manner.

The permutation group Γ acts on a set of points, $\mathcal{Z} \subset \mathbb{R}^n$, such that if $z \in \mathcal{Z}$ then $\pi(z) \in \mathcal{Z}$. The *orbit* of z under the action of the group Γ is the set of all elements of $\mathcal{Z}$ to which z can be sent by permutations in Γ, or $\text{orb}(z, \Gamma) \stackrel{\text{def}}{=} \{\pi(z) \mid \pi \in \Gamma\}$. Orbits can also be extended to variables. By definition, if e_j, the jth unit vector, is in $\text{orb}(\{e_k\}, \Gamma)$, then $e_k \in \text{orb}(\{e_j\}, \Gamma)$, i.e. the variable x_j and x_k share the same orbit. Therefore, the union of the orbits

$$\mathcal{O}(\Gamma) \stackrel{\text{def}}{=} \bigcup_{j=1}^{n} \text{orb}(\{e_j\}, \mathcal{G})$$

forms a partition of I^n, which we refer to as the *orbital partition* of Γ.

The stabilizer of $S \subseteq \mathcal{Z}$ is the subset of Γ that maps each element of S to an element in S, i.e., $\text{stab}(S, \Gamma) = \{\pi \in \Gamma \mid \pi(s) \in S \; \forall s \in S\}$.

We let $\mathcal{F}$ be the set of feasible solutions to BIP. The *symmetry group* of BIP, called $\mathcal{G}$, is the set of permutations of the variables that map each feasible solution onto a feasible solution of the same value.

$$\mathcal{G} \stackrel{\text{def}}{=} \{\pi \in \Pi^n \mid \pi(x) \in \mathcal{F} \text{ and } c^T x = c^T \pi(x) \quad \forall x \in \mathcal{F}\}.$$

Solutions x and $\pi(x)$, for any $\pi \in \mathcal{G}$, are called *isomorphic*. Note that the symmetry group acts on all elements of $\mathbb{R}^n$, not just the elements in $\mathcal{F}$. As a result, pairs of fractional solutions can be considered isomorphic.

Computing the symmetry group $\mathcal{G}$ of an integer program is likely far more difficult than solving the instance itself. In facting, simply testing if a permutation is in the problem instances symmetry group is NP-hard. This can be seen by reducing the feasibility problem for BIP to testing for membership in $\mathcal{G}$ (the same problem's symmetry group). If the BIP instance has n variables, then $n - 1$ tests can be made to determine

if the permutation mapping the element 1 to the element i, for $i \in \{2, \ldots, n\}$, while leaving all other elements unchanged, is in $\mathcal{G}$. If any permutation, π, is *not* in $\mathcal{G}$, then there must be a solution $x \in \mathcal{F}$, with $\pi(x) \notin \mathcal{F}$, meaning that there is a feasible solution to the BIP instance. If all permutations are in $\mathcal{G}$, then $\mathcal{G} = \Pi^n$. As such, the constraints $x_1 \geq x_2 \geq \cdots \geq x_n$ can be added to the problem formulation while preserving feasibility [4]. Since only $n+1$ integer solutions satisfy these constraints, each of these solutions can be tested for feasibility. If none are feasible, the BIP instance is infeasible.

Practical methods aimed at exploiting symmetries are forced to use a subgroup of the symmetry group that is found by examining the problem formulation. Let (A, b, c) be a formulation of BIP, i.e., $\mathcal{F} = \{x \in \{0, 1\}^v, Ax \geq b\}$. Given a permutation on the variables $\pi \in \Pi^v$ and a permutation on the constraints $\sigma \in \Pi^m$, let $A(\pi, \sigma) = P_\sigma A P_\pi$ be the matrix obtained by permuting the columns of A by π and the rows of A by σ, for permutation matrices P_σ and P_π. Applying the permutation to the right hand side vector gives $\sigma(b) = P_\sigma b$. The *formulation group* $\mathcal{G}(A, b, c)$ of BIP is the set of permutations

$$\mathcal{G}(A, b, c) \stackrel{\text{def}}{=} \{\pi \in \Pi^n \mid \exists\, \sigma \in Pi^m \text{ such that } A(\pi, \sigma) = A\,,\, \pi(c) = c, \text{ and } \sigma(b) = b\}.$$

A formulation group $\mathcal{G}(A, b, c)$ can be computed by using software (such as `nauty` [24] or `saucy` [6]) designed to compute the isomorphism group of a related graph. More details of the construction are available in [18]. Note that computing $\mathcal{G}(A, b, c)$ also generates a set of symmetries on the constraints. We call this set $\mathcal{C}(A, b, c)$. In the same way that O is the orbital partition of the variables, we let C be the orbital partition of the constraints. Note that the two constraints $a_1^T x \leq b_1$ and $a_2^T x \leq b_2$ share the same orbit if $b_1 = b_2$ and $a_1^T \pi(x) = a_2^T x$ for some $\pi \in \mathcal{G}$.

The formulation group $\mathcal{G}(A, b, c)$ is a subgroup of $\mathcal{G}$. In fact, there may be many formulations (A', b', c') of the problem, and the symmetries present in any formulation group may be used by symmetry-exploiting methods. This is important to consider when discussing cutting plane methods, as is done in Sect. 3. Generating a valid inequality and adding it to the formulation has the potential to reduce the size of a formulation group. However, because the cut does not remove any feasible solutions, the symmetry group does not change. Thus, it is appropriate to use the formulation group computed *before* adding the new cut as an approximation of the symmetry group.

Branch-and-bound trees for very small, but highly symmetric, BIP problems can be very large if the symmetry is ignored. The feasible region of a highly symmetric BIP may contain many thousands of equivalent optimal solutions. The presence of so many optimal solutions can make solving the BIP using branch-and-bound very difficult. One popular approach is to remove some of these equivalent solutions by adding symmetry breaking cuts.

Ideally, one would like to generate a set of cuts that remove all but one of each set of equivalent solutions. It is important, then, to determine which solution out of a set of equivalent solutions is chosen. This is typically done by using lexicographical ordering. The notation $z \preceq z'$ is used to denote lexicographic order. Because all variables are binary in BIP, we can simplify notation by referring to the solution $z \in \{0, 1\}^n$ as the

set $S = \{i | z_i = 1\}$ and $z' \in \{0, 1\}^n$ as the set $S' = \{i | z'_i = 1\}$. We have that $z \preceq z'$ if and only if $S \preceq S'$.

Unfortunately, restricting the feasible region to include only one element of each equivalent solution is likely to require an exponential number of constraints. This is evidenced by [21], where Luks and Roy show that an exponential number of constraints are needed to remove all non lexicographically minimal solutions from a problem's feasible region. Another approach is to use symmetry information to prune nodes in the branch-and-bound tree [23,28]. This is done by orienting the branch-and-bound tree, calling the branching that fixes a variable to one the *left branch* and the branch that fixes a variable to zero the *right branch*. The node a is *to the left of* node b if, at their first common ancestor, an ancestor of node a was formed by a left branch and an ancestor of node b was formed by a right branch. Ancestor nodes and their descendants are not compared in this way. This is illustrated in Fig. 1. A (possibly fractional) solution of node a is called *leftmost* at node a if there is no equivalent solution that is feasible at a node to the left of a. A node can be pruned if it can be shown that no integer feasible solution in the corresponding subproblem is leftmost. This can be a very effective technique, leading to a dramatic decrease in solution times. However, these branching techniques may still spend time solving unnecessary subproblems. Both isomorphism pruning and orbital branching only attempt to remove non-leftmost solutions from the feasible region by fixing variables and not by generating cuts. It is possible that the LP solution for a subproblem is not leftmost, and that removing all non-leftmost solutions may allow for the subproblem to be pruned by bound. This can be seen in Fig. 2. F, the dark shaded region in the feasible region, contains only non-leftmost solutions. If F is not removed from the feasible region, node b cannot be pruned by bound, however, removing F will allow pruning.

All, including fractional, non-leftmost solutions can be (impractically) removed by the following branching disjunction. Let node a in the branch-and-bound tree be defined by the set (F_0^a, F_1^a), where F_0^a is the set of variables fixed to zero at node a and F_1^a is the set of variables fixed to one. Let variable x_i be the variable chosen for branching. Suppose the branching disjunction

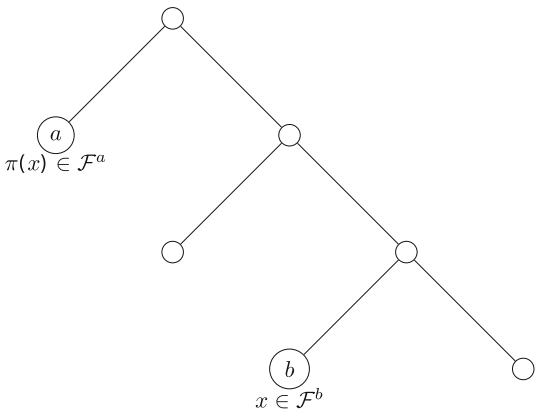

Fig. 1 Node a is the left of b, x is not leftmost

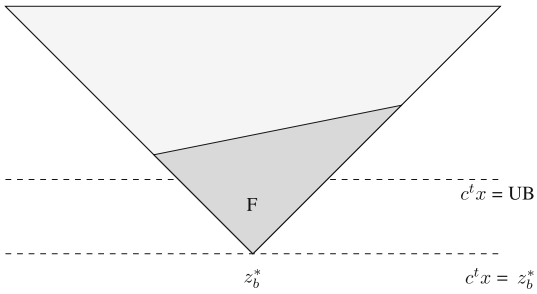

Fig. 2 Feasible region of b without symmetry breaking

$$x_i = 1 \vee \sum_{j \in \pi(F_1^a \cup \{i\})} x_j \leq |F_1^a| \quad \text{for every } \pi \in \mathcal{G} \tag{1}$$

is used. We call the child formed by fixing x_i to one the left child, and the child formed by adding the symmetry breaking constraints the right child. Note that the identity permutation fixes x_i to zero in the right child. It is easy to see that no solution in the right child is equivalent to a solution in left. If this branching disjunction is used at all ancestor nodes, then all feasible solutions in the right child will be leftmost. The difficulty with this branching method is in computing all of the symmetry breaking constraints. As will be shown in Table 1, $|\mathcal{G}|$ can be very large, reaching in the billions for (small) instances considered in this paper. The symmetry breaking method orbital branching [28] approximates this method, but replaces $\mathcal{G}$ with $\text{stab}(F_1^a, \mathcal{G})$.

3 Cutting planes and highly-symmetric problems

Cutting plane methods for general integer programs have been dramatically improved in the last decade and very effective implementations are now available in commercial and non-commercial solvers. Several types of general purpose cutting planes are known. Among them, Gomory *fractional* and *mixed integer* cuts, as well as *disjunctive cuts*, are often used to close the integrality gap, and help speed up the solution times of many integer programming problems. In the presence of symmetry, however, these methods can be inefficient. In this section, we review cutting plane methods and discuss why they tend to fail on highly symmetric problems. This is most evident when testing cutting plane methods on the most symmetric instances in the MIPLIB library, the Steiner triple systems (`sts`) problems.

Fischetti and Lodi [10] performed an exact optimization over the first *Chvàtal closure* of the natural $\{0, 1\} - LP$ formulation of `sts27` and `sts45`. They were not able to improve upon the initial lower bound. A similar result has been obtained in [3] when optimizing exactly over rank-1 *split cuts*, (equivalent to Gomory mixed integer cuts).

Balas et al. [34] showed that a standard implementation of Gomory's fractional cutting plane method (in which fractional Gomory cuts of any rank are generated) is able to close 50 % of the initial gap of `sts15`, but it fails to improve the initial bound of `sts27`. Interestingly, they also demonstrated that generating Gomory cuts to separate the lexicographically-largest optimal basic solution allowed them to solve to

Table 1 Size of shrunken LPs

Name	Group	Number of variables				Number of sets J			Number of constraints			
	Size	Orig	Lvl 1	Lvl 2	Lvl 3	Lvl 1	Lvl 2	Lvl 3	Orig	Lvl 1	Lvl 2	Lvl 3
Maximization												
cod103	40,874,803,200	1,024	6	27	297	6	50	771	1,024	12	185	5,546
cod105	40,874,803,200	1,024	6	27	297	6	50	771	1,024	12	185	5,546
cod83	92,897,280	256	5	18	123	5	30	281	256	10	110	1,964
cod93	1,857,945,600	512	6	23	200	5	39	468	512	10	143	3,342
Minimization												
codbt05	933,120	234	6	41	531	6	74	1,576	234	12	268	11,764
codbt15	1,866,240	486	12	121	3,519	12	274	12,004	486	24	1,040	92,704
codbt24	248,832	324	15	163	4,358	15	379	14,913	324	30	1,443	115,376
codbt33	62,208	216	16	159	3,438	16	373	11,636	216	32	1,430	89,944
codbt34	1,492,992	648	20	304	13,798	20	746	49,306	648	40	2,872	384,812
codbt42	27,648	144	15	121	1,836	15	279	5,984	144	30	1,068	45,998
codbt43	497,664	432	20	266	9,079	20	651	31,919	432	40	2,507	248,418
codbt52	276,480	284	18	184	4,229	18	444	14,382	284	36	1,710	111,300
codbt61	1,658,880	192	11	62	641	10	124	1,792	192	20	466	13,276
codbt71	23,224,320	384	12	82	1,178	12	178	3,491	384	24	674	26,852
codbt80	92,897,280	256	5	18	123	5	30	281	256	10	110	1,964
codbt90	1,857,945,600	512	6	23	199	5	39	468	512	10	143	3,342
cov1053	3,628,800	252	6	34	358	4	41	628	120	8	146	4,488
cov1054	3,628,800	252	6	34	358	5	55	910	210	10	197	6,586
cov1075	3,628,800	120	4	16	80	4	29	254	252	8	102	1,744

Table 1 continued

Name	Group	Number of variables				Number of sets J			Number of constraints			
	Size	Orig	Lvl 1	Lvl 2	Lvl 3	Lvl 1	Lvl 2	Lvl 3	Orig	Lvl 1	Lvl 2	Lvl 3
cov1076	3,628,800	120	4	16	80	4	28	233	210	8	98	1,594
cov1174	39,916,800	330	5	29	273	5	48	744	330	10	171	5,298
cov743	5,040	35	4	14	52	4	20	111	35	8	71	758
cov832	40,320	56	4	15	67	3	16	93	28	6	55	624
cov943	362,880	126	5	25	180	4	33	362	84	8	117	2,550
cov954	362,880	126	5	25	180	5	41	477	126	10	147	3,398
sts15	60	15	4	14	40	9	57	237	35	18	222	1,860
sts27	303,264	27	2	4	7	2	5	13	117	4	16	70
sts45	360	45	8	64	533	18	297	3,795	330	36	3,996	29,910
sts63	72,576	63	4	17	68	13	97	809	651	26	339	5,694
sts81	1,965,150,720	81	2	4	7	2	5	14	1,080	4	16	74

optimality for the first time (with a pure cutting plane algorithm) sts27. Even though sts27 contains only 27 variables (all of them binary), the cutting plane algorithm of [34] generated over 8,000 cuts in order to produce a tight integrality gap.

Disjunctive cuts also proved to be ineffective when applied to Steiner triple covering problems. Fischetti and Lodi [11] showed that no improvement of the initial gap has been accomplished on sts27 with ten rounds of disjunctive cuts. This result is surprising given the effectiveness of these cuts on other instances in the MIPLIB library.

While these sts problems may contain additional structure that make them notoriously difficult to solve, one explanation for the lack of success attained by these cutting plane methods is the high degree of symmetry present in these problems. Suppose one generated a valid inequality $\gamma x \leq \gamma_0$, referred to as (γ, γ_0), that removed the fractional solution x_{LP}. If the LP formulation contained a large degree of symmetry, it is likely that there is a solution equivalent to x_{LP}, $\pi(x_{LP})$, that satisfies the inequality (γ, γ_0). If such a $\pi(x_{LP})$ exists, adding the inequality (γ, γ_0) does not improve the LP relaxation. This is seen in [34], where the LP bound of the sts27 problem is displayed after successive rounds of Gomory cuts. Their results show that it is common for the LP relaxation of sts27 to remain unchanged after several rounds of cuts. The only way to guarantee that $\pi(x_{LP})$ is separated from the feasible region is to include the constraint $(\pi(\gamma), \gamma_0)$ to the LP formulation. Similarly, guaranteeing *all* solutions equivalent to x_{LP} are removed requires adding the set of inequalities $(\pi(\gamma), \gamma_o)$ for all π in $\mathcal{G}$. If $\mathcal{G}$ is very large (see Table 3 for the sizes of symmetry groups in common test instances), adding these inequalities may significantly affect the computational time needed to solve the LP relaxation. Fortunately, symmetry can be used to aid in the solving of the LP relaxation.

Theorem 1 *Let $f(x)$ be a convex function and X a convex set. There exists a solution $x_{avg} = argmin_{x \in X} f(x)$ with the property $x_{avg_i} = x_{avg_j}$ for all $i \in \text{orb}(\{e_j\}, \mathcal{G})$.*

Theorem 1 was proved in [13]. The intuition of the proof is that for any optimal solution x^*, $\pi(x^*)$ is both feasible and optimal. Taking the average of $\pi(x^*)$ for all π in $\mathcal{G}$ will give the desired $\bar{x}$. The motivation for Theorem 1 in [13] was for its application to semidefinite programming. This theorem is applied to linear programming in [4,27].

Corollary 1 *Given an LP formulation where O is the orbital partition of the variables with respect to the symmetry group of variables, $\mathcal{G}$, and C is the orbital partition of the constraints with respect to the symmetry group acting on the constraints, $\mathcal{C}$, one can construct an LP formulation containing $|O|$ (the number of non-isomorphic variables) many variables and $|C|$ (the number of non-isomorphic constraints) many constraints that has the same optimal objective value. We call this new LP formulation the* symmetry-shrunken *LP.*

The symmetry-shrunken LP is created by replacing all x_j terms in the problem formulation with y_i, where $i = \min(\text{orb}(\mathcal{G}, j))$. As a result of the definition of the formulation group, all constraints that are in the same constraint orbit will be identical. As an example, consider the LP

$$\begin{aligned} \min\ & x_1 + x_2 + x_3 \\ \text{subject to}\ & x_1 + x_2 \geq 1 \\ & x_2 + x_3 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_i \in [0,1] \quad \text{for } i = 1,\ 2,\ 3. \end{aligned} \tag{2}$$

All variables in (2) share the same orbit. Similarly, all constraints share the same constraint orbit. We can replace them with y_1 to give the shrunken LP

$$\begin{aligned} \min\ & 3y_1 \\ \text{subject to}\ & 2y_1 \geq 1 \\ & 2y_1 \geq 1 \\ & 2y_1 \geq 1 \\ & y_1 \in [0,1]. \end{aligned} \tag{3}$$

Note that all the constraints, which shared the same orbit, are now identical. Only one of the constraints is required for the formulation. The optimal solution to (3) has $y_1 = \frac{1}{2}$. This solution can be converted to an optimal solution to the original problem by letting $x_1 = x_2 = x_3 = y_1 = \frac{1}{2}$.

Corollary 1 can be used to solve the LP relaxation of highly-symmetric problems with a large number of constraints. However, separation may become difficult as the solution found in the shrunken LP may not be a basic solution in the original formulation. Finding a basic solution to the original formulation may require solving the original LP with *all* of the constraints. It is with this in mind that extended formulations of highly-symmetric problems are considered.

4 Extended formulations of highly-symmetric BIPs

As discussed in Sect. 3, it is not clear how to efficiently implement cutting plane methods for highly symmetric problems. Instead, we seek to obtain better bounds by exploiting Theorem 1. Because LP relaxations are much easier to solve when symmetry is present, we can consider much larger formulations.

Consider the Lovász–Schrijver relaxation of a $\{0,1\}$-integer program formed by the process defined by:

- **Step 1:** Generate the nonlinear system $(Ax - b)x_j \geq 0$ and $(Ax - b)(1 - x_j) \geq 0$ for each j in $\{0,\ \ldots,\ n\}$.
- **Step 2:** Substitute $y_{i,j}$, $i < j$, for each $x_i * x_j$ and $x_j * x_i$ with $i \neq j$, and x_i for every x_i^2. Let Y be a symmetric $(n+1)(n+1)$ matrix be defined by letting the vector $(1,\ x_1, \ldots,\ x_n)$ be both row and column 0. The entry $Y_{i,j}$ for $i < j$ is simply $y_{i,j}$. Let M be the convex set in $\mathbb{R}^{(\binom{n}{2}+n)}$ satisfying the inequalities in Step 1 with the additional restriction that Y be positive semidefinite.
- **Step 3:** Project M onto the x-space.

The Lovász-Schrijver Relaxation can be much stronger than the traditional LP relaxation. Also, Step 3 is not necessary, as optimizing a linear function over M is a semidefinite program (SDP). One downside of solving the SDP is that the SDP formulation is much larger than the original formulation, containing $\binom{n}{2}$ many additional variables. Recently, there has been interesting work in using the symmetry to reduce the number of variables in the SDP relaxation [7–9,32]. Note that lifting the LP formulation to M preserves symmetry. For every permutation π in the formulation group of the LP, we can construct a permutation for the SDP, π^+ by assigning $\pi^+(x)_i = \pi(x)_i$ and $\pi^+(y)_{i,j} = y_{\pi(i),\pi(j)}$. As M is a convex set, we can use Theorem 1 to add the constraints $x_i = x_{\pi^+(i)}$ and $y_{i,j} = y_{\pi^+(i),\pi^+(j)}$ for every π^+ a symmetry of M. The common strategy in SDP is to use these equalities to help construct a smaller SDP whose optimal value is equal to the full SDP. This section outlines a strategy similar to that used by the SDP community for solving large extended formulations.

In [30], Sherali and Adams describe a technique for generating a hierarchy of formulations of $\{0, 1\}$ integer programming problems that improve the quality of the linear relaxation. The rank d Sherali–Adams closure can be created by:

- **Step 1:** For every J_0 and J_1 subsets of $(1, \ldots, n)$ with $J_0 \cap J_1 = \emptyset$ and $|J_0| + |J_1| = d$, generate the nonlinear system $(Ax - b)(\prod_{i \in J_0} x_i)(\prod_{j \in J_1}(1 - x_j)) \geq 0$.
- **Step 2:** Substitute the binary variable x_S in place of every monomial $\prod_{i \in S} x_i^{d_i}$, call the set of solutions in this space M_d.
- **Step 3:** Project M_d onto the x-space, let P_d be the resulting polyhedron.

It can be shown [30] that $P_i \supset P_j$ for i less than j and that P_n equals the convex hull of $\mathcal{F}$. While the Lovász-Schrijver relaxation is guaranteed to be at least as strong as the Sherali–Adams level-1 formulation, there are some advantages to using Sherali–Adams. Namely, larger formulations that dominate the SDP formulation can be created and there is a wide variety of software available to solve linear programs (as opposed to semidefinite programs). There has been little work computing bounds associated with Sherali–Adams relaxations of levels greater than one. However, [5] finds that the level-1 Sherali–Adams bound reduced the integrality gap of several of the smaller MIPLIB instances by an average of 39 %.

Similar to the Lovász-Schrijver relaxation, the Sheral-Adams formulation can contain a large number of variables and constraints. In fact, the number of variables grows exponentially with d. However, as with the SDP case, the symmetry from the original formulation can easily be lifted to the extended formulation. As a result, for highly-symmetric problems, Corollary 1 can be used to create a substantially smaller LP formulation whose objective value is equal to the one found by optimizing over M_d (or P_d).

4.1 Constructing shrunken LPs

The naive way of constructing the level-d shrunken LP is to generate the entire extended formulation by first using the process described by the Sherali–Adams procedure. The constraints $y_S = y_{\pi(S)}$ for all π in $\mathcal{G}^d$, where $\mathcal{G}^d$ is the group of symmetries of the extended formulation induced by $\mathcal{G}$, can be added, then the LP can be processed. This

approach may not be computationally tractable for even small values of d, as each extended formulation contains more than $\binom{n}{d}$ many variables. Because of symmetry, for each constraint $ax \geq b$, there may be many combinations of sets J_0 and J_1 whose resulting nonlinear constraints $(ax-b)(\prod_{i\in J_0} x_i)(\prod_{j\in J_1}(1-x_i)) \geq 0$ are symmetric. Keeping this in mind, many redundant constraints can be excluded by considering Theorem 2.

Theorem 2 *Given constraint $a^T x \geq b$ in the original formulation, for any π in* stab$(a, \mathcal{G})$*, the linearized constraint generated by $(J_0,\ J_1)$ will be symmetric, with respect to $\mathcal{G}^d$, to that generated by $(\pi(J_0),\ \pi(J_1))$. Similarly, the set of constraints generated by multiplying $ax \geq b$ by all appropriate J_0 and J_1 terms will be symmetric to those generated using $\pi(a)^T x \geq b$ for all π in $\mathcal{G}$.*

Proof First, we show that all terms in the linearized constraint generated by $(\pi(J_0),\ \pi(J_1))$ and $a^T x \geq b$, $\texttt{rlt}_\pi$, can be mapped onto a term in the linearized constraint generated by (J_0, J_1) and $a^T x \geq b$, $\texttt{rlt}$.

The nonlinear constraint generated by J_0 and J_1 is:

$$\sum_{i=1}^{n} a_i x_i \left(\prod_{j\in J_0} x_j \prod_{k\in J_1} [1-x_k] \right) \geq b \left(\prod_{j\in J_0} x_j \prod_{k\in J_1} [1-x_k] \right). \tag{4}$$

Expanding out the $\prod_{k\in J_1}[1-x_k]$ term in (4) gives the constraint:

$$\sum_{i=1}^{n} \sum_{C\subseteq J_1} a_i (-1)^{|C|} \prod_{j\in J_0\cup C\cup\{i\}} x_j \leq \sum_{C\subseteq J_1} b(-1)^{|C|} \prod_{j\in J_0\cup C} x_j \tag{5}$$

Linearizing (5) gives $\texttt{rlt}_\pi$:

$$\texttt{rlt} \overset{\text{def}}{=} \sum_{i=1}^{n} \sum_{C\subseteq J_1} a_i (-1)^{|C|} x_{J_0\cup C\cup\{i\}} \geq \sum_{C\subseteq J_1} b(-1)^{|C|} x_{J_0\cup C} \tag{6}$$

Similarly, $\texttt{rlt}_\pi$, the linearized constraint generated by $(\pi(J_0), \pi(J_1))$ is:

$$\texttt{rlt}_\pi \overset{\text{def}}{=} \sum_{i=1}^{n} \sum_{C\subseteq \pi(J_1)} a_i (-1)^{|C|} x_{\pi(J_0)\cup C\cup\{i\}} \geq \sum_{C\subseteq \pi(J_1)} b(-1)^{|C|} x_{\pi(J_0)\cup C} \tag{7}$$

To show that $\texttt{rlt}$ is symmetric to $\texttt{rlt}_\pi$, we need to show that π is such that $a_{\texttt{rlt}}\pi(x) = a_{\texttt{rlt}_\pi}\pi(x)$. To do this, we will show that π maps each term of $\texttt{rlt}$ onto a unique term in $\texttt{rlt}_\pi$.

Consider the term $a_i(-1)^{|C|}x_{J_0\cup C\cup\{i\}}$ for some $i \in \{1, \ldots, n\}$ and some $C \subseteq J_1$. The permutation π maps $x_{J_0\cup C\cup\{i\}}$ to $x_{\pi(J_0)\cup\pi(C)\cup\{\pi(i)\}}$. We have that $\pi(C) \subseteq \pi(J_1)$, so we know the term $x_{\pi(J_0)\cup\pi(C)\cup\{\pi(i)\}}$ exists in $\texttt{rlt}_\pi$. Moreover, we know that the coefficient of $x_{\pi(J_0)\cup\pi(C)\cup\{\pi(i)\}}$ in $\texttt{rlt}_\pi$ is $a_{\pi(i)}$. Because $\pi \in$ stab$(a, \mathcal{G})$, it must

be the case that $a_i = a_{\pi(i)}$, meaning that each term on the left-hand side of (6) gets mapped to a term on the left-hand side of (7) with the same coefficient. A similar argument can be used to show that this is true for terms on the right-hand side as well.

If π is an onto mapping from terms in $\texttt{rlt}$ to terms in $\texttt{rlt}_\pi$, then π^{-1} is an onto mapping from terms in $\texttt{rlt}_\pi$ to terms in $\texttt{rlt}$, meaning that π not just an onto mapping, but a 1-1 mapping. This shows that the constraint $\texttt{rlt}$ is symmetric to the constraint $\texttt{rlt}_\pi$.

The second part of the theorem, that the set of constraints generated by multiplying $ax \geq b$ by all appropriate J_0 and J_1 terms will be symmetric to those generated using $\pi(a)^T x \geq b$ for all π in $\mathcal{G}$, follows by noting that $\pi(a)^T x = a^T \pi^{-1}(x)$. □

As a result of Theorem 2, the set of constraints can be restricted to only those generated by one representative of each constraint orbit. Also, J_0 and J_1 can be restricted such that J_0 is the lexicographically minimal representative of $\text{orb}(J_0, \text{stab}(a, \mathcal{G}))$ and J_1 is the lexicographically minimal representative of $\text{orb}(J_1, \text{stab}(J_0, \mathcal{G}) \cup \text{stab}(a, \mathcal{G})$. All such sets J_0 and J_1 can be found using techniques in non-isomorphic generation such as isomorphism pruning [22].

The resulting formulation will still contain variables y_S and $y_{\pi(S)}$ for some nontrivial π in $\mathcal{G}$. To generate the shrunken LP, all y_S terms can be replaced by y_{S^L}, where S^L is the lexicographically minimal representative of $\text{orb}(S, \mathcal{G})$. This process is formalized in Algorithm 1.

Algorithm 1 Generating a Level-d Shrunken LP

Initialize: $m =$ number of constraint orbits
for i **in** $\{1, \ldots, m\}$
Step 1: Choose constraint a_{i_1} from constraint orbit i.
Step 2: For every $J \subseteq \{1, \ldots, n\}$, with $|J| = d$ and $J \preceq \pi(J)\ \forall \pi \in \text{stab}(a_{i_1}, \mathcal{G})$
For every partition (J_0, J_1) of J with:
$J_0 \preceq \pi(J_0)$ for all $\pi \in \text{stab}(a_{i_1}, \mathcal{G})$
and $J_1 \preceq \pi(J_1)$ for all $\pi \in \text{stab}(J_1, \mathcal{G}) \cup \text{stab}(a_{i_1}, \mathcal{G}))$.
Step 3: Generate the nonlinear system $(a_{i_1} x - b)(\prod_{i \in J_0} x_i)(\prod_{j \in J_1} (1 - x_j)) \geq 0$
Step 4: Substitute the binary variable y_{S^L} in place of every monomial $\prod_{i \in S} x_i$ where $S^L \preceq S$ and there exists a $\pi \in \mathcal{G}$ with $S^L = \pi(S)$.

For highly-symmetric problems, the shrunken LP can be much smaller than the extended formulation, allowing us to compute the bound of the level-d relaxation for relatively large values of d. The strength of this result is demonstrated by computational results in Sect. 4.3.

4.2 Improving Sherali–Adams bounds

One option to further improve the Sherali–Adams bound is to add integrality constraints to some of the problem variables. Recall that in the shrunken LP, the variable y_{S^L} represents the average of all the variables $\{x_{\pi(S)} | \pi \in \mathcal{G}\}$. Thus, a new integer variable $Y_{S^L} = |\text{orb}(S, \mathcal{G})| y_{S^L}$, can be included in the shrunken LP. This presents two problems. Because $|\text{orb}(S, \mathcal{G})|$ can be very large, including the Y variables may add

numerical instability to the problem. Also, it is not clear that branching on Y_{S^L} is likely to be effective when $|\text{orb}(S, \mathcal{G})|$ is large, as it would only remove an interval of length $\frac{1}{|\text{orb}(S,\mathcal{G})|}$ from the range of possible values for y_{S^L}.

A class of constraints that are shown to be effective at improving the Sherali–Adams bounds exploits the relationship between the variables y_S, for $|S| = K$, and the original x_i variables. All of the problem instances examined in Sect. 4.3 were either covering or packing problems, where every variable in the original LP formulation has a coefficient of "1" in the objective function. A Sherali–Adams bound of z then implies (in the covering case) that at least z variables must take the value of 1 in any feasible solution. If at least z variables take the value of 1, then at least z choose 2 many binomial terms in the extended formulation must also take the value of one. For the level-k Sherali–Adams relaxation, the c onstraint

$$\sum_{\{S \in 2^S |\ |S|=i\}} y_S = \binom{\sum_{j=1}^{n} x_j}{i} \quad \text{for } i = 1 \ldots k+1 \tag{8}$$

is valid. In the shrunken LP, we can rewrite (8) as

$$\sum_{\{S^L |\ |S^L|=i\}} Y_{S^L} = \binom{\sum_{j=1}^{n} x_j}{i} \quad \text{for } i = 1 \ldots k+1. \tag{9}$$

We call the constraints in (9) *counting constraints*. Even though they are nonlinear, they can easily be linearized by introducing binary dummy variables d_i such that

$$\sum_{j=1}^{n} x_j = \sum_{i=1}^{n} d_i \tag{10}$$

$$d_i \geq d_{i+1} \ \forall i = 1, \ldots, n-1 \tag{11}$$

$$\sum_{\{S^L |\ |S^L|=i\}} Y_{S^L} = \sum_{j=1}^{n} \left(\binom{j}{i} - \binom{j-1}{i} \right) d_j \quad \text{for } i = 1 \ldots k+1. \tag{12}$$

Similar to the issues with branching on the y variables, introducing these constraints can cause numerical issues as a result of large values of Y_{S^L}. As a result, we use an exact LP solver.

When the shrunken LP is solved, the optimal solution, z_{LP}, is often fractional. Typically, in the minimization case, the best bound is then $\lceil z_{LP} \rceil$. However, due to the nonlinearity of the counting constraint, it may be possible to do better. For $j = \lceil z_{LP} \rceil$, fixing the dummy variable d_j to one and resolving may return an optimal solution with an objective value greater than j. In this case, d_{j+1} can be fixed to one and the shrunken LP can be resolved.

4.3 Computational results

In this section we compute the bound given by the level 1, 2, and 3 Sherali–Adams relaxations for highly symmetric BIP problems. The instances chosen for these tests come from combinatorial optimization problems with applications in coding theory and statistical design. Computations are given in Tables 1 and 3. The instances beginning with `cod` are used to compute maximum cardinality binary error correcting codes [20] and the instances whose names begin with `cov` are covering designs [25]. The instances beginning with `codbt` are used to computed minimum dominating sets in Hamming graphs. These include the famous "football pool problems". The instances `sts` are used to compute the incidence width of Steiner-triple systems [12]. Some classes of problems like `sts` and `cov` commonly include valid inequalities that are generated by looking at optimal solutions to smaller instances in the family. While these inequalities can strengthen the lower bounds considerably, they are not included in these computational results since our intent was to test the impact of Sherali–Adams relaxation on the original formulations. Including them would further strengthen the bound. Most instances are available on Francois Margot's website http://wpweb2.tepper.cmu.edu/fmargot/lpsym.html.

The shrinking procedure is very effective at reducing the size of the LP needed to compute the Sherali–Adams bound. Table 1 shows the number of variables in each relaxation. While the size of the corresponding LP still increases exponentially as the level increases, the reduction makes a noticeable difference in the size. Note that for only one instance, `sts45`, the number of variables in the level-2 relaxation is greater than the number of variables in the original problem. This is due to the fact that the size of the symmetry group for this instance is relatively small. Even in the level-3 relaxations, only one instance, `codbt34` contains more than 10,000 variables.

The number of variables in the level-k formulation corresponds to the number of nonisomorphic sets of size k with respect to the problem's symmetry group $\mathcal{G}$. The number of constraints generated by a single constraint $ax \leq b$ in the level-k formulation, on the other hand, corresponds to the number of nonisomorphic sets of size k with respect to the stabilizer of a with respect to $\mathcal{G}$. In the instances considered, all constraints were symmetric, so there was only one constraint orbit used in Algorithm 1. Even so, the extended formulations contained a large number of constraints. Table 1 gives the number of sets J used in Step 2 of Algorithm 1. Not only does the size of J increase considerably as the level increases; the size of J increases in proportion to the number of variables. Using all partitions (J_0, J_1) of J will result in generating $2^{|J|}$ many constraints. Limiting the constraints to only those that are lexicographically minimal with respect to $\text{stab}(J, \text{stab}(a))$ reduces the number of constraints generated by 5–10% for most instances. There is still a lot of redundancy in the set of constraints generated, though it is difficult to predict when such redundancies are likely to occur.

Table 2 shows the times required to generate and solve the shrunken LPs. Both CPLEX, version 12, and and QSopt_ex, a rational LP solver [2], were used to solve the LP problems. CPLEX was used to solve both the formulation with and without counting constraints, while QSopet_ex was used to solve only the formulations with the counting constraints. The barrier algorithm option was used for all CPLEX times. A "*" denotes that the LP problems were not solvable even when left to run for

Table 2 Time needed to generate and solving shrunken LPs

Name	Generating shrunken LP (S)			Solving Lvl 3 (S)		
	Lvl 1	Lvl 2	Lvl 3	CPLEX	QSopt_ex$^+$	CPLEX$^+$
cod103	17	235	32,708	<1	6	1
cod105	76	1,091	45,367	<1	<1	1
cod83	3	26	625	<1	<1	<1
cod93	5	71	2,381	<1	1	<1
codbt05	2	34	1,796	1	92	2
codbt15	6	232	24,790	71	*	*
codbt24	5	183	16,555	103	*	*
codbt33	4	115	8,832	54	16,694	317
codbt34	14	883	145,367	104,478	*	*
codbt42	3	63	3,380	12	2,945	27
codbt43	9	507	62,265	2,877	*	*
codbt52	7	244	19,684	76	*	*
codbt61	3	66	2,589	1	54	2
codbt71	8	218	11,750	5	414	212
codbt80	3	25	606	<1	<1	<1
codbt90	5	71	2,356	<1	2	<1
cov1053	3	48	2,029	1	10	1
cov1054	2	23	1,017	<1	11	1
cov1075	1	12	277	<1	<1	<1
cov1076	1	5	127	<1	<1	<1
cov1174	10	148	6,030	1	10	2
cov743	<1	1	17	<1	<1	<1
cov832	1	2	26	<1	<1	<1
cov943	2	7	210	<1	1	<1
cov954	1	8	231	<1	1	<1
sts15	1	1	13	<1	<1	<1
sts27	<1	1	2	<1	<1	<1
sts45	1	8	312	2	133	5
sts63	1	1	26	<1	<1	<1
sts81	1	2	9	<1	<1	<1

a day, whereas the "+" in the column label denotes the time needed to solve the shrunken LP with the counting constraints. Times required to solve level-1 and level-2 formulations were not reported, as all instances were solved in <1 s. As is shown by Table 2, the times required to generate each level of the Sherali–Adams formulation increases exponentially. There are several reasons for this. First, as shown in Table 3, the number of sets J that contain d elements and are lexicographically minimal with respect to stab(a), grows exponentially as d increases. Making things worse, every such J generates $2^{|d|}$ many constraints. The constraint generated by the partition (J_0, J_1)

Table 3 Bounds for Sherali–Adams

Name	Orig	SA bound			SA^+ bound			Optimal
	LP	Lvl1	Lvl2	Lvl3	Lvl1	Lvl2	Lvl3	
Maximization								
cod103	93	92	91	90	89	87	85	72
cod105	18	18	18	18	16	14	12	12
cod83	28	27	27	26	26	24	23	20
cod93	51	50	49	48	48	46	45	40
Minimization								
codbt05	23	23	23	24	24	25	26	27
codbt15	41	41	42	42	42	44	*	54
codbt24	30	30	31	31	31	32	*	36
codbt33	22	22	23	23	23	**24**	24	24
codbt34	54	55	56	56	56	58	*	72
codbt42	16	17	17	18	17	18	19	20
codbt43	40	40	41	41	41	42	*	48
codbt52	29	30	30	31	30	32	*	36
codbt61	22	22	23	23	23	**24**	24	24
codbt71	39	40	40	41	40	42	43	48
codbt80	29	30	30	31	30	31	**32**	32
codbt90	52	52	53	54	53	55	56	62
cov1053	12	13	13	13	13	14	14	17
cov1054	42	43	44	45	44	45	46	51
cov1075	12	13	14	14	14	15	16	20
cov1076	30	34	36	39	35	37	39	45
cov1174	10	10	10	10	11	12	12	17
cov743	9	10	11	11	10	11	**12**	12
cov832	10	10	10	10	10	10	**11**	11
cov943	21	21	22	22	21	22	23	25
cov954	26	27	28	28	28	28	29	30
sts15	5	7	7	8	7	7	8	9
sts27	9	12	13	15	13	13	16	18
sts45	15	20	21	24	20	22	24	30
sts63	21	27	30	34	27	31	37	45
sts81	27	35	38	44	40	40	48	61

Bold values indicate a zero gap

of J will contain $|J_1| * (nc + 1)$ many terms (where nc equals the number of non-zero coefficients of the constraint $ax \geq b$). For each term, $a_i x_S$, the lexicographically minimal set equivalent to S must be computed. The algorithm for computing the lexicographically smallest element is exponential with respect to the size of S. Note, however, that it may not be necessary to generate every constraint of the shrunken LP. Note from Table 3 that the number of multipliers J is always larger than the number

of variables in the shrunken LP, and, as a result, the number of constraints are much larger (more than a factor of 2^d) than the number of variables.

The bounds obtained by solving the levels 1 through 3 shrunken LPs are shown in Table 3. Bounds obtained by solving the original shrunken LP are labeled by "SA", while the bounds obtained with the counting constraints are labeled "SA$^+$". Unfortunately, because of the difficulty in solving LPs exactly, not all relaxations are solvable. A * denotes those that were unable to be solved. Surprisingly, even for the level-3 formulations, the improvements in the bound by Sherali–Adams are negligible. This is surprising given the results from [5], where the level-1 Sherali–Adams bound reduced the integrality gap by an average of 39 %. For these problems, the improvements for the level-1 bound were only 15 %. Even the level-3 bound was not as effective as the results from [5]. The bounds for two of the problems, `cov1174` and `cov832`, do not increase at all. Note, however, that the bounds did improve significantly for the `sts` problems. The counting constraints, however, can be helpful in improving the bound. Including the counting constraints reduced the gap by an extra 10, 20, and 30 percentage points in the level 1, 2, and 3 cases. Interestingly, in several instances the level-$k-1$ formulation with the counting constraints appears stronger than the level-k constraints without them. In fact, with the counting constraints, optimality is proven in 6 of the test instances (two at level 2 and four at level 3).

Note from Table 1 that both third level formulations for `sts27` and `sts81` contain just 7 variables. This is a result of the fact that both instances contain a small number of original variables and also have very large symmetry groups (relative to the number of variables). As a result, higher level formulation can be easily created. Table 4 gives the bounds for these two instances for levels 4 through 7. Optimality is proven in level 4 for `sts27`, while the integrality gap is nearly closed in the seventh level for `sts81`.

5 Using extended formulations in the branch-and-bound tree

Even with the addition of the counting constraints, the bound provided by the Sherali–Adams relaxation is unable to prove optimality for most of the problem instances. Many of the unsolved problems still have large integrality gaps. Considerable effort could be spent trying to compute the level-4 or even level-5 Sherali–Adams relaxation, but those relaxations seem unlikely to prove the optimality of many more problems. Even though the extended formulations are not likely to prove optimality by themselves, it may be beneficial to solve extended formulations at subproblems throughout the branch-and-bound tree. Unfortunately, these extended formulations are only computationally tractable due to the large amount of symmetry in the original problem formulation. Symmetry is quickly removed from the formulation as variables are fixed to one. It

Table 4 Lower bound provided for high level Sherali–Adams formulations

Instance	% lower bound from SA$^+$				Opt
	Lvl 4	Lvl 5	Lvl 6	Lvl 7	
sts27	16	18	18	18	18
sts81	53	53	56	57	61

is unlikely, then, that even the level-1 relaxation of subproblems will be solvable in an acceptable period of time. Despite these limitations, the extended formulations can still be a valuable tool in identifying prunable nodes in the branch-and-bound tree. The main reason for this is the fact that symmetry breaking constraints can be efficiently enforced in the Sherali–Adams formulation. In this section, we show the impact of the pruning on several test problems.

As described in Sect. 2, symmetry breaking algorithms such as isomorphism pruning and orbital branching identify nodes to be pruned or variables to be fixed by testing if all feasible solutions are non-leftmost. It may be the case, however, that the optimal LP solution of a subproblem is not leftmost, but its feasible region contains leftmost solutions. In this case, the subproblem is not pruned by either of these two methods. No attempt is made to remove the non-leftmost optimal solution from the feasible region, as it is not clear how to guarantee that the optimal solution is leftmost without adding an exponential number of constraints to the original formulation. This is not always the case, however, for the Sherali–Adams formulation. For all nodes in the branch-and-bound tree with at most k variables fixed, the level-k Sherali–Adams formulation can be used to strengthen the lower bound on the child nodes. This is done, in part, by removing non-leftmost solutions from the Sherali–Adams relaxation.

Suppose the right constraint in the symmetry-breaking disjunction described in (1) were added to the formulation. Note that $\sum_{i \in S} x_i \leq |S| - 1$ can be written as the nonlinear constraint $\prod_{i \in S} x_i = 0$. If S is such that $|S| \leq k + 1$, the (non-shrunk) level-k extended formulation will contain a variable y_S. Note that $|S| \leq k+1$ because nonlinear formulation of the level-k extended formulation contains monomials with degree less than or equal to $k+1$. The nonlinear formulation of the branching constraint can be useful in the extended formulation, as it simply implies that $y_S = 0$. Enforcing that $y_{\pi(S)} = 0$ for all $\pi \in \mathcal{G}$ can be done easily in the shrunken formulation by fixing y^L_S to 0.

To demonstrate the impact of the Sherali–Adams formulation on the branch-and-bound tree, we use the level-3 formulations for several of the test problems considered in Sect. 4.3 to prune nodes in the in the corresponding branch-and-bound trees. First, we solve each problem without using the Sherali–Adams formulation. The problems are solved using isomorphism pruning. At every node, the smallest-indexed variable is chosen for branching. For every node a in the branch-and-bound tree with $|F^a_1| \leq 3$, if variable x_i is chosen for branching, we fix the variable $y^L_{F^a_1 \cup \{x_i\}}$ to zero in the right child (the subproblem formed by fixing x_i to zero). Solving the Sherali–Adams relaxation with this constraint will provide a valid lower bound for the right child. While it would be valid to add the constraint $y^L_{F^a_1 \cup \{x_i\}} > 0$ to the Sherali–Adams formulation to compute the bound for the left branch, our exerience shows that adding this constraint is unlikely to make a difference. Because we are using level-3 formulations, we can only use the Sherali–Adams formulation to improve the lower bound on nodes who are children of nodes with only 3 variables fixed to on (and their ancestors).

Table 5 shows the number of child subproblems in the branch-and-bound tree generated by parents with exactly 3 variables fixed to one using isomorphism pruning (labeled "ISOP") and using with the Sherali–Adams formulation (labeled "ISOP+SA"). As the results show, the Sherali–Adams relaxation (with the counting

Table 5 Nodes in tree with $|F_1^a| = 4$

Problem	ISOP	ISOP + SA
cod83	3	2
cod93	6	5
codbt05	37	35
codbt90	15	12
cov943	15	12
cov954	8	2
cov1075	15	15
cov1076	11	6
cov1054	15	12
cov1053	60	57
sts45	8	2
sts63	4	1

constraints) can be used to prune nodes early on in the branch-and-bound tree. Most notably, six of the eight nodes in `cov954` and six of the eight nodes in `sts45` are pruned. Still, though, there are problems where few, if any, nodes are not able to be pruned. It is unclear why this method is effective for only some problems.

5.1 Case study: the football pool problem

The problems listed in Table 5 were chosen to demonstrate how the Sherali–Adams formulation can be used in the branch-and-bound tree. The problems themselves, however, are not sufficiently difficult to make this approach computationally effective. The large amount of time required to generate the Sherali–Adams formulation, combined with the time needed to solve each relaxation, makes this strategy useful only for exceptionally difficult problems. With that in mind, we apply these techniques to the football pool problem.

The football problem is a popular problem from coding theory [16], due to its application to gambling on football (soccer) games. A gambler wishes to bet on n football games. Each bet can be expressed as an n-digit ternary word, where the ith letter of the word describes the predicted outcome of game i ("0" if the home team wins, "1" if the home team loses, and "2" if the game ends in a tie). The gambler wins a large payout if he makes a bet that predicts all of the games, save one, correctly. The gambler would like to know the minimum number of bets required to guarantee that he wins the payout. This problem can be expressed as a covering problem on a hamming graph with 3^n vertices. The variable x_i is equal to one if the bet corresponding to the ternary expansion of i is placed. Each vertex represents a possible outcome, and two vertices are adjacent if their outcomes differ only in one game. Solving these problems is extremely difficult due to the presence of symmetry. For any feasible cover, merely permuting the letters in the corresponding ternary word will generate a new feasible solution. Permuting the meaning of each letter is also a symmetry. As such, the MILP

Table 6 Optimal values for the football pool problem

Problem size	1	2	3	4	5
Optimal value	1	3	5	9	27

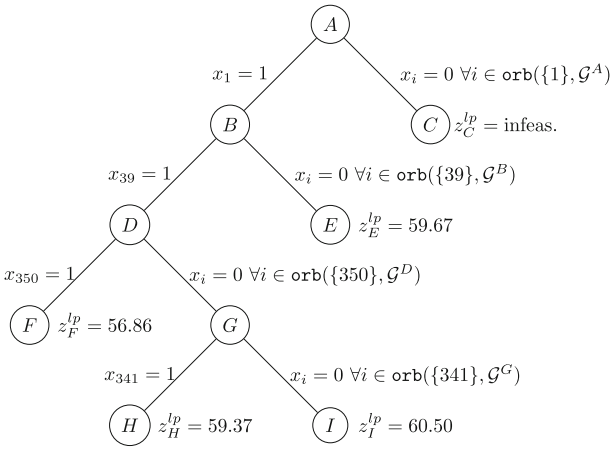

Fig. 3 Traditional branch-and-bound tree for `f6`

formulation of the football pool problem in n games contains $3^n n!$ many symmetries. The instances with known optimal values are given in Table 6.

The smallest unsolved problem is where one is betting on only six games, which we will refer to as `f6`. The best known feasible solution is 73 [33]. Recently, after tremendous computational effort, a lower bound of 71 has been proved [19]. This required more than 2 centuries of computational effort. The best known lower bound prior to this was 65 [26]. Using branch-and-bound to solve this problem has, to date, been unsuccessful. [19] reports that after processing 500,000 nodes in the branch-and-bound tree, CPLEX v9.1 is only able to improve the lower bound from 56.08 to 58.

Unfortunately, the level-3 Sherali–Adams formulation of `f6` proved to be too large to generate. However, the level-2 formulation can be generated in under 2 h. The lower bound associated with the level-2 formulation is 66, a slight improvement in the bound provided by [26]. In addition, this formulation can be very useful in minimizing the number of subproblems early in the branch-and-bound tree.

In this experiment we use the level-2 Sherali–Adams relaxation to prune difficult subproblems in the branch-and-bound tree. Figure 3 shows the early portion of the branch-and-bound tree generated using orbital branching. In this tree, only subproblem C is pruned. Figure 4 shows how the level-2 Sherali–Adams bound can be computed for each subproblem in the tree. For example, the bound for subproblem E can be formed by fixing $y^L_{1,30}$ to zero (we do not enforce the strict inequalities associated with a left branch). Using the Sherali–Adams relaxation, nodes C, E, and I can be pruned by bound. As mentioned, node C is easily pruned when using just orbital branching. However, subproblems E and I are considerably more difficult to solve. After 10 h of computing time (each) CPLEX was unable to solve either subproblem, but found a

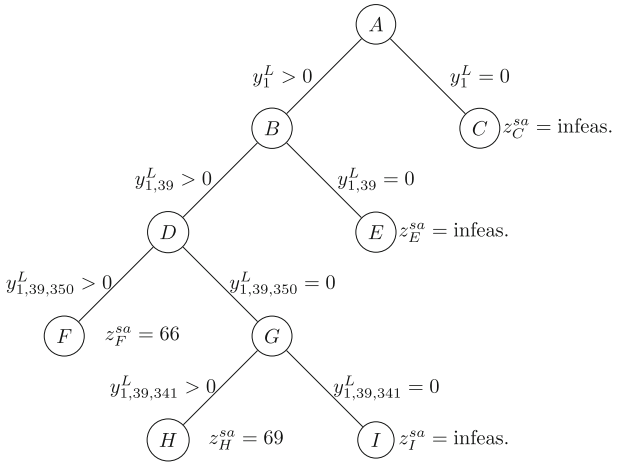

Fig. 4 Sherali–Adams branch-and-bound tree for `f6`

lower bound of 67.93 for subproblem E and 66.21 for subproblem I. Generating the level-2 relaxation requires approximately 2 h, and solving for the bound at each node takes approximately 10 s each. This shows a clear benefit to using the Sherali–Adams relaxation to prune nodes.

Future work In this paper we focused on generating the entire description of the shrunken Sherali–Adams formulation. It might be computationally beneficial to only generate a subset of the variables and constraints. For example, generating and solving the level-3 formulation of `f6` is likely impossible. However, the bound on the right child of subproblem F may be improved by adding constraints to the level-2 formulation. Suppose that Algorithm 1 was used for $J = \{1, 39, 350\}$. This process will generate some of the level-3 constraints containing variables of the form $y_{1,39,350,i}$. Adding these constraints to the level-2 formulation should improve the relaxation while still leaving the corresponding LP solvable. While the resulting formulation will not be as strong as the level-3 formulation, it can be used to improve the bound for the right child of F.

6 Conclusions

In this paper, we show how to exploit symmetry when solving linear relaxations of highly symmetric BIPs. When it is present, we can construct LP problems that have objective values identical to that of Sherali–Adams relaxations. These LPs can provide tighter bounds than the original formulation, and in seven of the test instances they can be used to prove optimality. The relaxations can also provide efficient ways to implement symmetry breaking constraints and help to prune nodes early in the branch-and-bound process. This is demonstrated by pruning some extremely difficult subproblems of the football pool problem. This work shows that solving Sherali–Adams relaxations can be computationally feasible, and can be used to aid in the solving of the BIP.

Acknowledgments The author would like to thank Ricardo Fukasawa, Jeff Linderoth, Sven Leyffer, and the anonymous referees for their helpful comments about this work.

References

1. Adams, W., Guignard, M., Hahn, P., Hightower, W.: A level-2 reformulationlinearization technique bound for the quadratic assignment problem. Eur. J. Oper. Res. **180**(3), 983–996 (2007)
2. Applegate, D., Cook, W., Dash, S., Espinoza, D.: QSopt ex. http://www.dii.uchile.cl/daespino/ESolverdoc/main.html (2007). Accessed 7 Oct 2012
3. Balas, E., Saxena, A.: Optimizing over the split closure. Math. Program. Ser. B **113**(2), 219–240 (2008)
4. Bödi, R., Herr, K., Joswig, M.: Algorithms for highly symmetric linear and integer programs. Math. Program. **137**, 1–26 (2011)
5. Bonami, P., Minoux, M.: Using rank-1 lift-and-project closures to generate cuts for 0–1 mips, a computational investigation. Discrete Optim. **2**(4), 288–307 (2005)
6. Darga, P.T., Liffiton, M.H., Sakallah, K.A and Markov, I.L.: Exploiting structure in symmetry detection for CNF. In: Design Automation Conference (DAC), pp. 530–534 (2004)
7. De Klerk, E., Maharry, J., Pasechnik, D.V., Richter, B., Salazar, G.: Improved bounds for the crossing numbers of km, n and kn. SIAM J. Discrete Math. **20**, 189–202 (2006)
8. De Klerk, E., Pasechnik, D.V., Schrijver, A.: Reduction of symmetric semidefinite programs using the regular *-representation. Math. Program. B **109**(2–3), 613–624 (2007)
9. De Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Math. Program. Ser. B **122**(2), 225–246 (2010)
10. Fischetti, M., Lodi, A.: Optimizing over the first chvátal closure. Math. Program. Ser. B **110**(1), 3–20 (2007)
11. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Math. Program. **128**, 205–230 (2011)
12. Fulkerson, D.R., Nemhauser, G.L., Trotter, L.E.: Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triples. Math. Program. Study **2**, 72–81 (1974)
13. Gatermann, K., Parrilo, P.: Symmetry groups, semidefinite programs, and sums of squares. Pure Appl. Alg. **192**, 95–128 (2004)
14. Ghoniem, A., Sherali, H.: Defeating symmetry in combinatorial optimization via objective perturbations and hierarchical constraints. IIE Trans. **43**(8), 575–588 (2011)
15. Hahn, P.M., Zhu, Y.-R., Guignard, M., Hightower, W.L., Saltzman, M.J.: A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. INFORMS J. Computi. **24**(4), 202–209 (2012). (spring)
16. Hämäläinen, H., Honkala, I., Litsyn, S., Östergård, P.: Football pools-a game for mathematicians. Am. Math. Mon. **102**(7), 579–588 (1995)
17. Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. In IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization. pp. 74–88. Springer (2007)
18. Liberti, L.: Reformulations in mathematical programming: automatic symmetry detection and exploitation. Math. Program. **131**, 273–304 (2012)
19. Linderoth, J., Margot, F., Thain, G.: Improving bounds on the football pool problem via symmetry reduction and high-throughput computing. INFORMS J. Comput. **21**, 445–457 (2009)
20. Litsyn, S.: An updated table of the best binary codes known. In: Pless, V.S., Huffman, W.C. (eds.) Handbook of Coding Theory. volume 1, pp. 463–498. Elsevier, Amsterdam (1998)
21. Luks, E.M.: Permutation groups and polynomial-time computation. In: Larry, F., William, M.K. (eds.) Groups and Computation, Volume 11 of American Mathematical Society, DIMACS Series. pp. 139–175, (DIMACS, 1991) (1993)
22. Margot, F.: Pruning by isomorphism in branch-and-cut. Math. Program. **94**, 71–90 (2002)
23. Margot, F.: Exploiting orbits in symmetric ILP. Math. Program. Ser. B **98**, 3–21 (2003)
24. McKay, B.D.: Nauty User's Guide (Version 1.5). Australian National University, Canberra (2002)
25. Mills, W.H., Mullin, R.C.: Coverings and packings. In: Dinitz, J.H., Stinson, D.R. (eds.) Contemporary Design Theory: A Collection of Surveys, pp. 371–399. Wiley, New York (1992)
26. Östergård, P., Wassermann, A.: A new lower bound for the football pool problem for six matches. J. Comb. Theory Ser. A **99**(1), 175–179 (2002)

27. Ostrowski, J.: Symmetry handling in mixed-integer programming. In: Cochran, J., Cox, L., Keskinocak, P., Kharoufeh, J., Smith, C. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley (2010)
28. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Math. Program. **126**, 147–178 (2011)
29. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Solving large steiner triple covering problems. Oper. Res. Lett. **39**(2), 127–131 (2011)
30. Sherali, H., Adams, W.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discrete Math. **3**(3), 411–430 (1990)
31. Sherali, H., Smith, C.: Improving discrete model representations via symmetry considerations. Manage. Sci. **47**, 1396–1407 (2001)
32. Shrijver, A.: New code upper bounds from the Terwilliger algebra. IEEE Trans. Inf. Theory **51**, 2859–2866 (2005)
33. Wille, L.T.: The football pool problem for 6 matches: A new upper bound obtained by simulated annealing. J. Comb. Theory Ser. A **45**(2), 171–177 (1987)
34. Zanette, A., Fischetti, M., Balas, E.: Lexicography and degeneracy: can a pure cutting plane algorithm work? Math. Program. **130**, 153–176 (2011)