Discrete Applied Mathematics 222 (2017) 109–123

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

# Orbital shrinking: Theory and applications

Matteo Fischetti [a], Leo Liberti [b,*], Domenico Salvagnin [a], Toby Walsh [c]

[a] DEI, Università di Padova, Italy
[b] CNRS LIX, Ecole Polytechnique, 91128 Palaiseau, France
[c] NICTA and UNSW, Sydney, Australia

ARTICLE INFO

*Article history:*
Received 27 January 2016
Received in revised form 8 January 2017
Accepted 20 January 2017
Available online 21 February 2017

*Keywords:*
Mathematical programming
Constraint programming
Discrete optimization
Symmetry
Relaxation
MINLP

ABSTRACT

We present a method, based on formulation symmetry, for generating Mixed-Integer Linear Programming (MILP) relaxations with fewer variables than the original symmetric MILP. Our technique also extends to convex MINLP, and some nonconvex MINLP with a special structure. We showcase the effectiveness of our relaxation when embedded in a decomposition method applied to two important applications (multi-activity shift scheduling and multiple knapsack problem), showing that it can improve CPU times by several orders of magnitude compared to pure MIP or CP approaches.



## 1. Introduction

Branch-and-Bound (BB) type methods often become very slow when the solution set is symmetric [11,25], due to the exploration of many symmetric subtrees. Given a Mathematical Programming (MP) formulation, we distinguish the automorphism group of its solution set (called the *solution group*) and the group of variable symmetries fixing the formulation (called the *formulation group*). The latter is usually defined as the group of variable index permutations keeping the objective function invariant and permuting the order of the constraints [3,23]. It is very easy to show that the formulation group is a subgroup of the solution group.

Finding a universal technique for determining the solution group automatically would imply knowing the solution set *a priori*, which would make the optimization problem moot. On the other hand, various techniques for finding the formulation group of a Constraint Satisfaction Program (CSP) and of a MP have been proposed in the literature [2,19,20,37]. The most efficient methods reduce to the graph isomorphism problem, which can be solved in practice using tools such as `nauty` [27].

Once some symmetries are known, they can be exploited in a variety of ways. In Constraint Programming (CP) and Mixed-Integer Programming (MIP), a common technique consists in trying to make some of the symmetric solutions infeasible by:

- adjoining Symmetry-Breaking Constraints (SBC) to the original formulation [20,21,46];
- using a clever branching strategy in constraint propagation [10] or in BB (e.g. isomorphism pruning [23,24] or orbital branching [30,31]).

In Semidefinite Programming (SDP), due to the fact that decision variables are matrices, symmetry can be exploited very naturally through group representation theory. This yields formulations with fewer variables (in fact, the variable matrix becomes block-diagonal) but having the same optimum [8].

* Corresponding author.
*E-mail addresses:* matteo.fischetti@unipd.it (M. Fischetti), liberti@lix.polytechnique.fr (L. Liberti), domenico.salvagnin@unipd.it (D. Salvagnin), toby.walsh@nicta.com.au (T. Walsh).

http://dx.doi.org/10.1016/j.dam.2017.01.015
0166-218X/

A different approach is proposed in [2], where solving an Integer Linear Program (ILP) with a highly transitive solution group is essentially reduced to a line search in a lattice. The proposed method is very innovative, but most practically occurring ILPs have groups that are very far from being transitive. A generalization of this approach which aims to relax the requirement for high transitivity is given in [14]. Although applicability remains limited, this technique was used in solving the instance `toll-like` in the MIPLIB2010 library [18], which was previously unsolved.

We propose another approach, called *Orbital Shrinking* (OS), for exploiting symmetry in MILP and certain subclasses of Mixed-Integer Nonlinear Programming (MINLP). Orbital shrinking is a relaxation technique: given a MIP $P$ and a subgroup $G$ of its formulation (or solution) group, it replaces each orbit of variables by a single variable. Therefore, OS produces compact MIP relaxations. If $G$ is transitive, and hence has only one orbit, the resulting MIP is trivial, because it has only one variable. At the other extreme, if $G$ is the trivial group, then there are as many orbits as there are variables, and the relaxation is the same as the original MIP.

To solve problems exactly, we employ the OS relaxation (OSR) in a general purpose decomposition framework, which we apply to two real-life applications: multi-activity shift scheduling and multiple knapsack problems. OS decomposition naturally provides a new way for designing hybrid MIP/CP decompositions: our computational results show that the resulting method can be orders of magnitude faster than pure MIP or CP approaches.

The outline of the paper is as follows. In Section 1.1, we review some main results on symmetry groups in the context of optimization problems. Then, in Section 2, we present orbital shrinking, and show that it yields a relaxation of the original problem. In Section 3 we analyze differences and similarities between OS and core point algorithms. In Section 4 we describe a general decomposition framework based on orbital shrinking, while in Sections 5 and 6 we specialize the general framework to multi-activity shift scheduling and multiple knapsack problems, also reporting computational results. Conclusions are finally drawn in Section 7.

We assume the reader is familiar with mixed-integer programming, constraint programming and basic group theory. The present paper extends and is based on the preliminary results presented in [7,42,43], by the same authors.

### 1.1. Some notation and terminology

Let $P$ be an arbitrary MINLP of the form

$$\min f(x) \tag{1}$$

$$\forall i \in C \qquad g_i(x) \leq 0 \tag{2}$$

$$\forall j \in J \qquad x_j \in \mathbb{Z} \tag{3}$$

where $J \subseteq [n] = \{1, \ldots, n\}$ is the subset of integer variables. Without loss of generality, the objective function $f(x)$ is assumed to be convex. For a point $x' \in \mathbb{R}^n$ and a subset $V \subseteq [n]$, we let $x'[V]$ be the subsequence of $x'$ indexed by $V$.

We consider the formulation group $G_P$ of $P$, containing the set of permutations $\pi \in S_n$ (the symmetric group of order $n$, which acts naturally on the variable indices) that leave the formulation of $P$ unchanged, except for a possible reordering of the constraints. The practical applicability of this definition extends to Linear Programs (LP) and MILPs. With MINLPs, we restrict our attention to functional forms which are closed with respect to the usual operators $(+, -, \times, \div, (\cdot)^a)$ and unary functions (log, exp). These expressions are easily represented by trees, and whole MINLP formulations can be represented by suitable Directed Acyclic Graphs (DAG) [1]. $G_P$ is then obtained as a restriction of the automorphism group of this DAG to the set of variable indices of $P$ [20]. $G_P$ can be computed by means of any graph isomorphism package such as Nauty [28] or Saucy [17]: these both implement backtracking algorithms which are exponential-time in the worst case, but which are sufficiently fast in practice to be of use.

Any subgroup $G$ of $S_n$ partitions the set of variables into equivalence classes called *orbits* via its natural action: two variable indices $i, j$ are in the same class if there is $g \in G$ such that $g(i) = j$. We denote by $\Omega_G$ the *orbital partition* of the action of $G$ on $[n]$. We remark that, by definition, integer and continuous variables cannot be permuted with each other, so each orbit contains only integer or only continuous variables. Constraints of $P$ are themselves partitioned into equivalence classes, called *constraint orbits*: in particular, two constraints are in the same orbit if and only if one is mapped into the other (because of reordering) when some variable permutation $\pi \in G$ is applied. Finally, given a subset $I \subseteq [n]$, the *point-wise stabilizer* $G[I]$ of $G$ with respect to $I$ is the subgroup of $G$ consisting of permutations $\pi$ such that $\pi(i) = i$ for all $i \in I$.

## 2. Orbital shrinking relaxation

The OSR with respect to a subgroup $G$ of $S_n$ could be best described as "formulation modulo $G$", as it replaces entire orbits by single variables. In this section, we will describe how to construct the OSR of a given optimization problem $P$, and show that this is indeed a relaxation of the original problem.

The first step is to classify variables and constraints according to their incidence and (non)linearity/convexity. Specifically, the group $G$ defining the OSR will be taken with respect to partitions $(V_1, V_2)$ and $(C_1, C_2, C_3)$ of the variables and constraints of $P$ respectively, that satisfy the following conditions: