Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

BENJAMIN HILLER, TOM WALTHER

Improving branching for disjunctive polyhedral models using approximate convex decompositions

The authors thank the DFG for their support within project A04 in CRC TRR154 and the BMBF Research Campus Modal (fund number 05M14ZAM) and ICT COST Action TD1207 for additional support.

ZIB Report 17-68 (November 2017)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Improving branching for disjunctive polyhedral models using approximate convex decompositions

Benjamin Hiller Tom Walther

November 30, 2017

Abstract

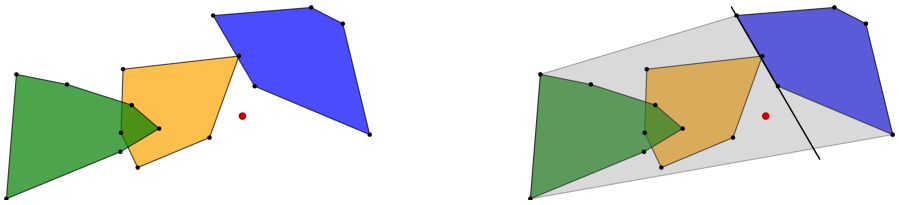
Disjunctive sets arise in a variety of optimization models and much research has been devoted to obtain strong relaxations for them. This paper focuses on the evaluation of the relaxation during the branch-and-bound search process. We argue that the branching possibilities (i.e., binary variables) of the usual formulations are unsuitable to obtain strong bounds early in the search process as they do not capture the overall shape of the the entire disjunctive set. To analyze and exploit the shape of the disjunctive set we propose to compute a hierarchy of approximate convex decompositions and show how to extend the known formulations to obtain improved branching behavior.

1 Introduction

Disjunctions of (polyhedral) sets are a frequently occuring structure in many optimization models. They arise in a variety of ways. Obviously, they are necessary to capture switching and decision aspects in the problem domain and to model the effect of these decisions. Additionally, they may result from model reformulations, for instance to model piecewise linear functions as approximations of nonlinear functions [22] or due to reformulations to improve the strength of the model [3].

For global optimization, the strength of the relaxation is very important. As the strongest relaxation of a nonconvex set is its convex hull, there has been research to determine models for this convex hull based on models for the nonconvex set. In the case of disjunctive polyhedral sets (i.e., unions of polytopes), the so called *convex hull reformulation* [2] describes the convex hull in terms of the inequalities of the original polytopes. The resulting model features one binary variable for each original polytope that is used to "select" this polytope. To obtain strong relaxations for models involving several disjunctions that have to be fulfilled simultaneously, another disjunction of usually more polytopes can be derived as described in [3]. The convex hull reformulation of this disjunction often yields a stronger relaxation.

The outlined results focus on obtaining a strong *initial* relaxation, i.e., the relaxation of the root node in a branch-and-bound search tree. To successfully solve the model using branch-and-bound algorithms, it is also important that the relaxation becomes stronger while diving into the search tree. A nice property of the convex hull reformulation in this respect is that after excluding



(a) A union of polytopes.

(b) Convex hull reformulation.

Figure 1: Relaxations of linear disjunctive constraint representing the union of polytopes.

a polytope (by fixing the corresponding binary variable to 0), the remaining model yields the convex hull of the remaining polytopes. Thus the relaxation is as good as possible when branching on the binary variables from the convex hull reformulation.

However, as we will detail in Section 2, in cases of disjunctions with many polytopes this relaxation may tighten slowly during branching and thus yields large search trees. Consider the example illustrated in Figure 1: The disjunctive polyhedral set consists of three polytopes, two of which determine the convex hull. The point outside the union of the three polytopes represents a hypothetical solution of the relaxation that is infeasible for the union. Suppose this relaxed solution arises as a convex combination from the yellow and the blue polytope. The solver may then decide to branch on the binary variable representing the yellow polytope. While in the 1-branch, infeasibility of the point is immediately detected, the relaxation does not improve at all in the 0-branch. If, however, the solver would have the possibility to branch on the cut shown in Figure 1b, the relaxation would be tight enough to cut off the point in *both* branches.

The example shows that in general, one can benefit from exploiting knowledge about the *shape* of the disjunctive set to obtain tighter relaxations during branching. Before outlining our approach, we should mention that the above example can easily be resolved by applying a "basic step" as described in [3] to the disjunctive sets given by the union of the three polytopes on the one hand and the set containing just the relaxation solution on the other hand. This reformulation is possible (and worthwhile) if two disjunctive sets are known to often appear jointly in one model, such that a refined model combining them is a useful submodel. In contrast, the ideas presented here focus on a single disjunctive set, assuming that the disjunctive set itself is an interesting submodel to be used in larger models.

Our contribution In this paper, we propose to employ *approximate convex decomposition methods* to derive suitable information and guidance for branching. Let $\mathcal{P}$ be a disjunctive polyhedral set, i.e., $\mathcal{P} = \cup_{i=1}^{n} P_i$, where each $P_i \subseteq \mathbb{R}^d$, $1 \leq i \leq n$, is a polytope. The idea is to cover $\mathcal{P}$ with polytopes $C_1, \ldots, C_m \subseteq \mathbb{R}^d$ such that

- $\mathcal{P} \subseteq \bigcup_{i=1}^{m} C_i$,
- $\mathcal{P} \cap C_i$ is "approximately convex" w.r.t. to some measure for nonconvexity,

The set of components $C_1, \ldots, C_m$ is called an *approximate convex decomposition* of $\mathcal{P}$. We employ the volume difference between the convex hull of the set and the set itself to measure nonconvexity.

As the implementation complexity of known methods for obtaining approximate convex decompositions increases significantly with d, we restrict ourselves to the case $d = 2$ to investigate the potential of this approach. In particular, we adjust the approximate decomposition method from [14] to our purposes as it is simple to implement. Moreover, as it successively refines a decomposition by cutting the component C_i with the highest nonconvexity, it naturally yields a decomposition hierarchy.

The results from this procedure may be used in several ways to reformulate or extend the original model. In general, the polytopes constituting $\mathcal{P}$ are not disjoint and some parts of their descriptions are not relevant for the boundary of $\mathcal{P}$. Hence the size of an approximate convex decomposition in terms of number of polytopes and facets might be smaller, in particular if a certain approximation error is accepted. Thus an approximate convex decomposition provides a potentially smaller, but coarser replacement for the original disjunctive set. More importantly, the formulation for this reformulated disjunctive set can be extended by a model of the decomposition hierarchy that encodes the corresponding refinement in terms of additional binary variables. In this way, branching may take into account information about the overall shape of the disjunctive set as suggested in our motivating example. Finally, it is possible to include these refined models in the original MILP or MINLP formulation to provide improved branching opportunities.

Related work The results mentioned above are commonly refered to as Disjunctive Programming techniques [2, 3, 4]. In recent years, these results have been extended to the case of convex and nonconvex sets, see e.g., [5, 13, 20].

The *exact* convex decomposition of nonconvex polyhedra for the case $d = 2, 3$ has already been studied in the eighties [6]. The problem is known to be NP-hard and there are lower bounds on the number of required convex components. In the context of computer graphics and modeling, several methods for obtaining approximate convex decompositions have been proposed [14, 15, 1, 17, 16] These methods have been designed for the 2- or 3-dimensional case; some of them [1, 17] also work, in principle, for arbitrary d, at the expense of more complex data structures. Most of these measure nonconvexity using the length of a shortest geodesic path from a point in $\partial\mathcal{P}$ to $\partial\text{conv}\mathcal{P}$ or some approximation of thereof. This measure is not suitable for assessing the strength of a relaxation. Hence we use the (relative) volume difference as our measure of nonconvexity.

Our computational results are based on polyhedral models arising in gas network optimization [12, 10]. For an overview on related work in this area we refer to [21].

The remaining paper is structured as follows. In Section 2 we review the well-known convex hull reformulation and argue why this formulation may suffer from poor bounds during the branch-and-bounds process. Section 3 proposes means to obtain improved overall models for a disjunctive polyhedral set based on approximate convex decompositions. These improved models are validated computationally in Section 4.

2 Models for disjunctive polyhedral sets in the branch-and-bound solution process

A *disjunctive polyhedral set* $\mathcal{P}$ is a set that can be written as the union of polytopes, i.e., $\mathcal{P} = \cup_{i=1}^{n} P_i$, where each $P_i \subseteq \mathbb{R}^d$, $1 \leq i \leq n$, is a polytope. The condition $x \in \mathcal{P}$ is often expressed as the *disjunctive constraint*

$$\bigvee_{i=1}^{n} \{x \in P_i\}. \tag{1}$$

In the following, we will assume that each polytope P_i is given by its inequality description $P = \{x \in \mathbb{R}^d \mid A^i x \leq b^i\}$.

We will now discuss how this constraint can be formulated as a MILP.

2.1 Modeling disjunctive polyhedral sets

One well-known way of formulating constraint (1) as a MILP is the so-called big-M approach [19]. However, is it known that in general the big-M approach yields weak LP relaxations that strictly contain the convex hull of $\mathcal{P}$.

There is a formulation that yields the convex hull of $\mathcal{P}$ and thus the tightest possible convex relaxation. This formulation is known as *convex hull reformulation* [3]. This formulation uses, for each polytope P_i, a binary variable s_i and continuous so-called disaggregation variables $v_i \in \mathbb{R}^d$ and can be stated as follows:

$$x = \sum_{i=1}^{n} v_i, \tag{2a}$$

$$A^i v_i - b^i s_i \leq 0 \qquad 1 \leq i \leq n, \tag{2b}$$

$$\underline{x} s_i \leq v_i \leq \overline{x} s_i \qquad 1 \leq i \leq n, \tag{2c}$$

$$\sum_{i=1}^{n} s_i = 1 \tag{2d}$$

$$s_i \in \{0, 1\} \qquad 1 \leq i \leq n. \tag{2e}$$

In this model, $\underline{x}, \overline{x}$ are finite lower and upper bounds for the variable vector x (component-wise), which exist as all P_i are bounded.

For brevity, we will refer to the entire model (2) for a disjunctive polyhedral set $\mathcal{P}$ using the notation $\mathsf{convhull}\,[(P_i)_{1 \leq i \leq n};\ x;\ s]$.

2.2 Evolution of the relaxation during branch-and-bound search

We will now show that in cases of disjunctive polyhedral sets consisting of many polytopes, branching only on the binary variables corresponding to each polytope leads to slow convergence of the search process. We do this by studying two didactic examples. Throughout we assume that the disjunctive polyhedral sets are modeled via the convex hull reformulation (2), which yields the convex hull of any subset of the polytopes, too.

k	5	10	20	30	40	50	60	70	80	90	100
#nodes	1	1	71	115	139	199	217	261	293	331	355

Table 1: Number of search tree nodes necessary to show infeasibilty for various values of k.

2.2.1 A feasibility problem

Let $P_1, \ldots, P_k \subset \{(x, y) \in \mathbb{R}^2 \mid x < 0\}$ and $P_{k+1}, \ldots, P_{2k} \subset \{(x, y) \in \mathbb{R}^2 \mid x > 0\}$ be polytopes and consider the disjunctive polyhedral set $\mathcal{P} = \cup_{i=1}^{2k} P_i$. The task is to prove $0 \notin \mathcal{P}$.

As $0 \in \text{conv}\{P_i, P_j\}$ for any $1 \leq i \leq k$ and $k+1 \leq j \leq 2k$, it is necessary to exclude at least k polytopes via branching. In terms of the formulation (2), this means that at least k binary variables need to be fixed in the branch-and-bound search tree. However, the resulting search tree is not a full binary tree, but a degenerate one, as the selection of a polytope immediately shows infeasibility of the corresponding subproblem. Note that branching on "$0 \in P_1, \ldots, P_k$" or "$0 \in P_{k+1}, \ldots, P_{2k}$" obviously yields the desired conclusion in one step.

We conducted some experiments to investigate the solution behavior for this simple disjunctive polyhedral set using the MILP solver SCIP 4.0[18]. When feeding the formulation (2) together with the constraint $x = 0$ into the solver, infeasibility is determined immediately in the preprocessing phase. The reason for this are the powerful preprocessing techniques of modern solvers. In this case, infeasibility is established by a technique known as *probing*. In fact, in this setting probing effectively performs a "basic step" between the disjunctive polyhedral sets $\mathcal{P}$ and $\{0\}$. This means that two distinct components of the model are combined to conclude infeasibility. As we want to study the behavior of formulations of disjunctive polyhedral sets in isolation, we disabled probing. Moreover, we disabled *strong branching* for the same reason.

Table 1 shows the number of search tree nodes required by SCIP for various values k and randomly generated[1]polytopes. The number of nodes in the search tree grows roughly linear in k.

2.2.2 An optimization problem

To show the implications of the limited branching opportunities for optimization, we consider the following example. Similarly to the preceding construction, let $P_1, \ldots, P_k \subseteq [-1, 1] \times [0, k]$ and $P_{k+1}, \ldots, P_{2k} \subset [0, k] \times [-1, 1]\}$ be polytopes and consider the disjunctive polyhedral set $\mathcal{P} = \cup_{i=1}^{2k} P_i$. Moreover, let x_{ref} be an infeasible reference point near the barycenter of conv$\mathcal{P}$ (see Figure .

The task is to find a feasible point $x \in \mathcal{P}$ with minimal distance to x_{ref}, i.e.,

$$\min_{x \in \mathcal{P}} ||x - x_{\text{ref}}||_2.$$

[1]The exact generation procedure does not matter much. We used the following setup. Each polytope arises as the convex hull of 20 points chosen uniformly at random from $[-1, 1] \times [-1, 1]$. Polytope i, $1 \leq i \leq k$, is shifted in the direction of the x-axis by $S_i := 1 + \sum_{j=1}^{i} o_i$, where each o_i is chosen uniformly at random from $[0, 2]$. Polytopes i, $k+1 \leq i \leq 2k$, are generated analogously but shifted in the other direction.

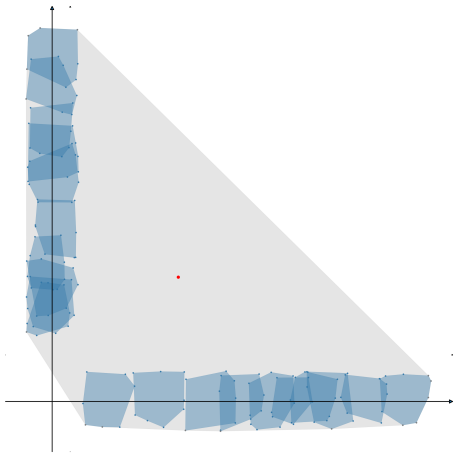


Figure 2: An instance of the example optimization problem for $k = 10$.

When thinking about the constraint $x \in \mathcal{P}$ as being part of a bigger model, this objective can be viewed as a proxy for the remaining model which prefers a solution close to x_{ref}. Note that we chose a nonlinear objective as a linear objective is always minimized on the convex hull of $\mathcal{P}$.

The reference point lies within the convex hull of many subsets of the polytopes. Accordingly, many polytopes need to be excluded in order to obtain a positive lower bound for the distance. In contrast to the last example, the 1-branch selecting a polytope is not infeasible, but yields the much simpler subproblem of determining the minimum distance between the polytope and x_{ref}. Note that branching on "$x \in P_1, \dots, P_k$" or "$x \in P_{k+1}, \dots, P_{2k}$" immediately yields a positive lower bound for the distance between $x \in \mathcal{P}$ with minimal distance to x_{ref}.

Using an analogous procedure to the last experiment, we generated problem instances for various values of k and solved them with SCIP 4.0. With probing and strong branching enabled, SCIP solves the models with only a few nodes. However, this is due to heavy collaborative work of the entire presolving machinery, involving several restarts to finally obtain a reasonable tight bound to solve the instance. For probing and strong branching disabled, Figure 3 shows the evolution of the lower bounds with the number of branching nodes visited. As expected, many nodes are required to eventually obtain good bounds.

In the two examples discussed, a significant amount of branching is required to solve the models. In fact, these models can only be solved rapidly as the remaining subproblem in one branch direction is trivial. In more complex models, this is likely not the case, and the branching effort to deal with several disjunctions multiplies. In both examples there is an obvious significantly better branching opportunity that cannot be realized by the solver due to the formulation used. In the following, we will address this issue by proposing a method to reformulate the disjunctive polyhedral set in order to provide additional branching opportunities that yield tighter relaxations.

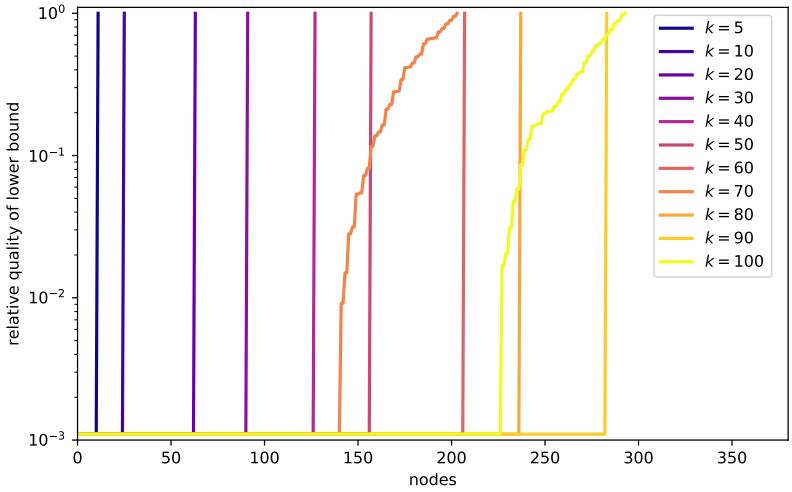


Figure 3: Evolution of the quality of the lower bound relative to the optimal solution for various values of k. In all but two cases the lower bound is actually 0 until the last node is finished. For the remaining two cases $k \in \{70, 100\}$ it still takes many nodes to raise the lower bound above 0 and to close the gap.

3 Approximate convex decompositions for disjunctive polyhedral constraints

In this section, we describe in detail how to obtain approximate convex decompositions and how to use them to construct models that facilitate improved branching during the solution process.

3.1 Computing a hierarchy of approximate convex decompositions

Our starting point is the algorithm proposed in [14] to construct approximate convex decompositions of a disjunctive polyhedral set $\mathcal{P} = \cup_{i=1}^{n} P_i$. We chose this algorithm because it has a generic structure and it is relatively simple to implement, thus lending itself to experiments. Moreover, as it successively refines a decomposition by cutting a single component C_i, it naturally yields a decomposition hierarchy.

The first step in order to apply the algorithm is to compute an explicit representation of $P := \partial\mathcal{P}$, i.e., the boundary of our disjunctive polyhedral set. This is a well-known and well-solved problem; in our implementation, we are using the freely available Python package SHAPELY [8] for this operation. In the general case, P is a nonconvex polygon and may have holes.

The algorithm from [14] takes a (nonconvex) polygon P as input and recursively splits P into two (nonconvex) components such that some measure of nonconvexity is reduced in every recursion. The algorithm uses the length of a shortest geodesic path from any vertex x of P to a vertex of the con-

vex hull of P as a nonconvexity measure for the vertex x. We will denote this distance as $\mathsf{spdist}_P(x)$, using the convention $\mathsf{spdist}(P) := \max\{\mathsf{spdist}_P(x) \mid x \text{ is a vertex of } P\}$. This nonconvexity measure is local, i.e., it assigns a nonconvexity value to every point $x \in \partial P$. This property is necessary for the algorithm as a single vertex x^* maximizing $\mathsf{spdist}_P(x)$ is selected as the basis for splitting the polygon. This splitting is accomplished by a procedure called $\mathsf{Resolve}(P, x^*)$, which searches a connection from x^* to some other vertex of P that is used to split P into two subpolygons P_1 and P_2 such that $P = P_1 \cup P_2$. If there are any holes in the original polygon, these are considered infinitely nonconvex and thus resolved first. For a more detailed description of the algorithm, we refer to the original paper [14].

We propose the extension of this algorithm outlined in Algorithm 1 to compute a decomposition hierarchy for P that captures approximate convex decompositions. The idea is to "record" the successive refinements due to the Resolve procedure in a tree $T = (V, E)$ with root $v_0 \in V$. For each node v, there is a nonconvex polygon P_v represented by v; the root node v_0 represents the original polygon P. Figure 4 shows a decomposition hierarchy as produced by a typical run of Algorithm 1.

Algorithm 1: Construncting a decomposition hierarchy.

Input: A (non-convex) polygon P and a nonconvexity tolerance τ_{sp}
Output: A decomposition hierarchy $T = (V, E)$ with root node v_0, family of polygons for each node $(P_v)_{v \in V}$

$V \leftarrow \{v_0\}$, $E \leftarrow \emptyset$, $P_{v_0} \leftarrow P$
$Q \leftarrow \{v_0\}$
while $Q \neq \emptyset$ **do**
  Pop node v from the queue Q.
  Compute $\mathsf{spdist}(P_v)$.
  if $\mathsf{spdist}(P) > \tau_{\mathsf{sp}}$ **then**
    Let $x^* \in \partial P$ be a point realising $\mathsf{spdist}(P)$.
    $V \leftarrow V \cup \{v_1, v_2\}$, $E \leftarrow E \cup \{(v, v_1), (v, v_2)\}$
    $P_{v_1}, P_{v_2} \leftarrow \mathsf{Resolve}(P_v, x^*)$
    $Q \leftarrow Q \cup \{v_1, v_2\}$
  end
end

The decomposition hierarchy $T = (V, E)$ computed by Algorithm 1 gives rise to a variety of approximate convex decompositions in the following way. First we associate with every node v corresponding to a polygon P_v a *convex component* C_v via $C_v := \mathrm{conv}(P_v)$. Note that C_{v_0} is the convex hull of the original disjunctive set $\mathcal{P}$. A *covering node set* $L \subseteq V$ is a set of nodes that are the leaves of a subtree of T rooted at v_0. For a non-leaf node v with child nodes v_1 and v_2 we have, due to the Resolve-procedure, that $P_v = P_{v_1} \cup P_{v_2}$ and thus $P_v \subset C_{v_1} \cup C_{v_2}$. Thus for a covering node set $L \subset V$ we have $\mathcal{P} \subseteq \bigcup_{v \in L} C_v$ and hence the family $(C_v)_{v \in L}$ is an approximate convex decomposition of $\mathcal{P}$.

A nice additional advantage of considering approximate convex decompositions is that often distinct convex components are disjoint and thus lead to disjoint subproblems in the search tree.

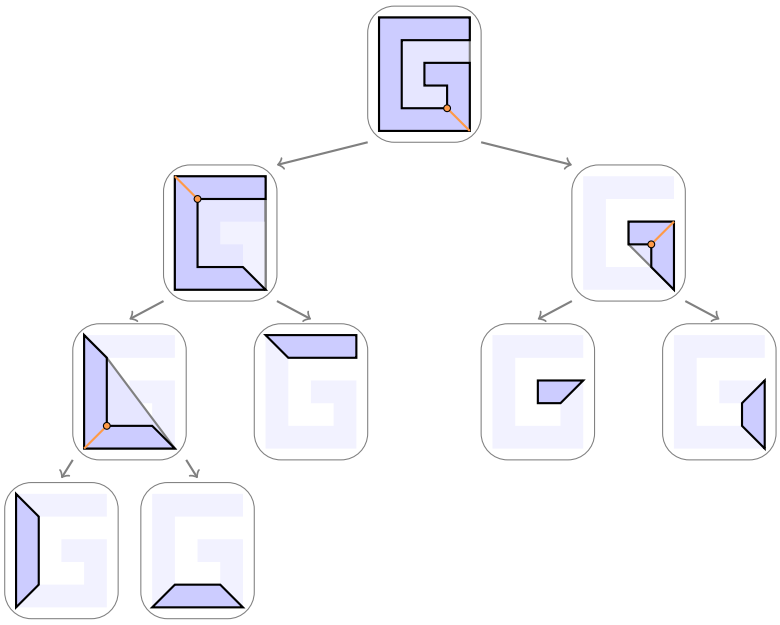


Figure 4: Example of a decomposition hierarchy produced by the ACD algorithm. The edge used for cutting into two components is shown in orange.

3.2 Using approximate convex decompositions as replacement model

By definition, any approximate convex decomposition $(C_1, \dots, C_m)$ of $\mathcal{P}$ can be used to relax (1) to

$$\bigvee_{i=1}^{m} \{x \in C_i\}. \tag{3}$$

To judge the quality, i.e., tightness, of this relaxation, we use a measure for nonconvexity that provides a global view rather than the local measure that is used within Algorithm 1. A natural and intuitive choice is based on the ratio of volumes

$$\frac{\operatorname{vol}\left(\mathcal{P} \setminus \left(\bigcup_{i=1}^{m} C_i\right)\right)}{\operatorname{vol}\mathcal{P}} = \frac{\operatorname{vol}\left(\bigcup_{i=1}^{m} C_i\right)}{\operatorname{vol}\mathcal{P}} - 1. \tag{4}$$

We propose the following simple algorithm for selecting an approximate convex decomposition for a given maximum convexification error τ_{vol} as definded in (4). Starting with the covering node set $L = \{v_0\}$, we successively replace a node v by its sucessors v_1 and v_2 as long as $\frac{\operatorname{vol} C_v}{\operatorname{vol} P_v} - 1 > \tau_{\mathsf{vol}}$. Note that by construction, L remains a covering node set and thus yields the approximate convex decomposition $(C_v)_{v \in L}$. While this procedure may not yield the minimal decomposition (in terms of number of components) with the specified convexification error (larger errors for some components cannot be outweighed by smaller errors for others), it guarantees that the global threshold is satisfied.

3.3 Using the hierarchy of approximate convex decompositions to guide branching

Replacing the original disjunctive set $\mathcal{P}$ by an approximate convex decomposition improves branching only if the number of polytopes is reduced. This is due to the fact that no additional branching opportunities are introduced and the effects discussed in Section 2.2 apply.

To incorporate additional branching opportunities, we can combine the convex hull reformulation for $\mathcal{P}$ with the convex hull reformulation of an (potentially coarse) approximate convex decomposition $\mathcal{C} := (C_1, \dots, C_m)$ in one model:

$$\mathsf{convhull}\left[(P_i)_{1\le i\le n};\ x;\ s^{\text{orig}}\right], \tag{5a}$$
$$\mathsf{convhull}\left[(C_i)_{1\le i\le m};\ x;\ s^{\text{approx}}\right]. \tag{5b}$$

Branching on the additional binary variables s^{approx} introduced for $\mathcal{C}$ exploits knowledge of the shape of $\mathcal{P}$ encoded in the inequalities for each component C_i. In this way, branching on these binary variables early provides a tighter relaxation quickly, while the disjunctive constraint for $\mathcal{P}$ ensures equivalence to the original model.

Obviously it is possible to use several approximate convex decompositions in addition to the original model at the same time to capture a hierarchy of refinements in the overall model. However, this introduces a lot of additional disaggregation variables and constraints that can be avoided.

Consider again the decomposition tree $T = (V, E)$. We say that a covering node set V^2 is a *refinement* of a covering node set V^1 if any node $v \in V^1$ is covered by a subset $V^2(v) \subseteq V^2$. Note that the subsets $V^2(v)$ are disjoint. Consider two approximate convex decompositions $\mathcal{C}^2 := (C^2_u)_{u\in V^2}$ and decomposition $\mathcal{C}^1 := (C^1_v)_{v\in V^1}$ arising from covering node sets V^1, V^2 where V^2 is a refinement of V^1. Due to this property, the two levels of the hierarchy can be expressed simultaneously using the binary variables of the finer approximate convex decomposition $\mathcal{C}^2$ and additional binary variables for $\mathcal{C}^1$. This observation yields the model:

$$\mathsf{convhull}\left[(P_i)_{1\le i\le n};\ x;\ s^{\text{orig}}\right], \tag{6a}$$
$$\mathsf{convhull}\left[(C^2_u)_{u\in V^2};\ x;\ s^2\right], \tag{6b}$$
$$\sum_{u\in V^2(v)} s^2_u = s^1_v \qquad v \in V^1, \tag{6c}$$
$$s^1_v \in \{0,1\} \qquad v \in V^1. \tag{6d}$$

Note that branching on the coarse-level binary variable s^1_v either removes all corresponding components from $\mathcal{C}^2$ from the disjunctive constraint for $\mathcal{C}^2$ ($s^1_v = 0$) or restricts the disjunctive constraint to those $\mathcal{C}^2$-components corresponding to C^1_v ($s^1_v = 1$). In both cases, the convex hull reformulation for $\mathcal{C}^2$ yields the same relaxation as if $\mathsf{convhull}\left[(C^1_v)_{v\in V^1};\ x;\ s^1\right]$ was included in the model as well.

Of course, model (6) may be extended to include more than two levels. If the finest level is actually an exact decomposition of $\mathcal{P}$ or the convexification error is acceptably small, constraint (6a) may be omitted.

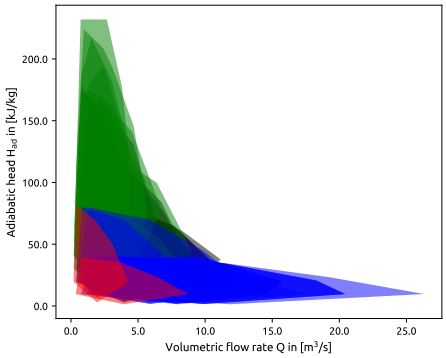
4 Computational Results

Motivated by our work in gas network optimization [12, 10], we consider polyhedral disjunctive sets modeling gas compressor stations. Each polytope of this disjunctive set corresponds to a *configuration* employing a subset of the compressor machines in the station in a particular way. The union of these configuration polytopes describes the set of feasible operating points of the entire compressor station. We have used publicly available data from the GasLib [11], in particular from the instance GasLib-582-v2. As the compressor stations available in GasLib are not as complex as real-world stations, we have merged compressorStation_{2|3|4} into an aggregated and hence more complex station for our testing purposes. Our aggregated compressor station consists of 6 compressor machines that can be operated in 42 different configurations (see Figure 5). We have uniformly sampled 5000 points within the convex hull of the configuration polytopes. The task is to decide whether each of these points is a feasible operating point.

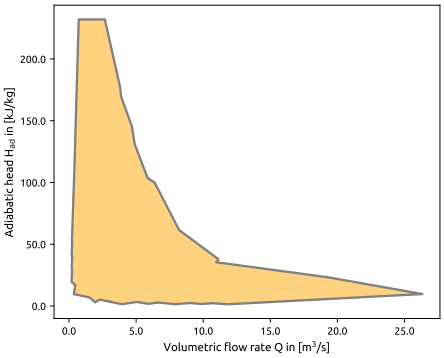
Running Algorithm 1 on this disjunctive sets yields an approximate convex decomposition with the number of components depending on the specified nonconvexity threshold. The first two columns of Table 2 indicate the tradeoff between the number of components and the resulting convexification error. The convex hull itself (i.e., one component) has a very high convexification error of more than 100%. It is evident that the refinement of the decomposition quickly reduces the error. In particular, the errors are just around 1% or less for approximate convex decompositions with only 4, 5 or 8 components. Thus the shape of the entire disjunctive set is effectively captured by the successive decompositions. Even an exact convex decomposition, i.e., one with convexity error 0.0, consists of only 13 convex components. Hence, in this case, the number of binary variables is reduced from 42 to 13 without losing accuracy.

For each approximate convex decomposition, we created an optimization model to check the feasibility of a operating point using the convex hull reformulation. We solved this model using SCIP 4.0[18] to determine which of the 5000 potential operating points are feasible for the approximate convex decomposition and which are not. The results are shown in the remaining columns of Table 2, including the overall time needed for checking the points. The running times are significantly lower for approximate convex decompositions with fewer components. The final stages of the decomposition (i.e., with 8 and 13 components) yield almost the same feasibility result as the original model, but require 25% (15%) less running time.

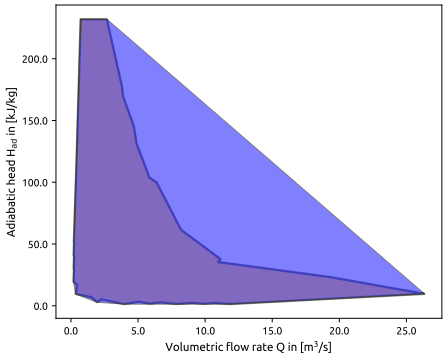
In addition to the MILP model resulting from the disjunctive set, we also investigated using an approximate convex decomposition to improve the full MINLP model for the compressor station. This full MINLP model [7, 9] features additional variables coupled by several nonlinear equations and inequalities. We strengthened the full model by the convex hull reformulation for each approximate convex decomposition computed before. This is valid since the configuration polytopes have been obtained as a very accurate approximation for the full MINLP model [9]. Table 3 shows the results of solving these models, again using SCIP 4.0. It can be seen that the number of instances that are feasible or infeasible remains almost the same across all model variants; we attribute the small deviations to numerical precision issues. For the feasible instances, adding the model for the approximate convex decomposition does not affect



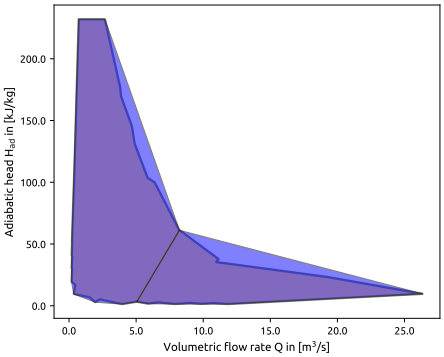
(a) 6 machines, 42 configurations.



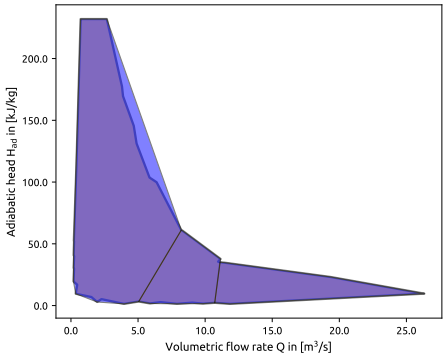
(b) Aggregated characteristic diagram.



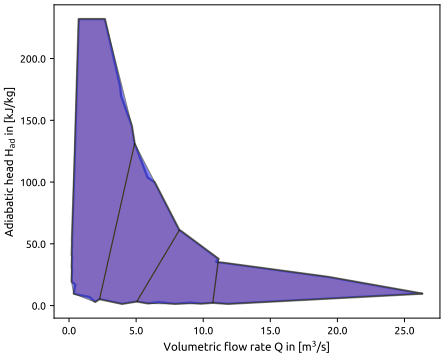
(c) 1 components, error = 102.51%.



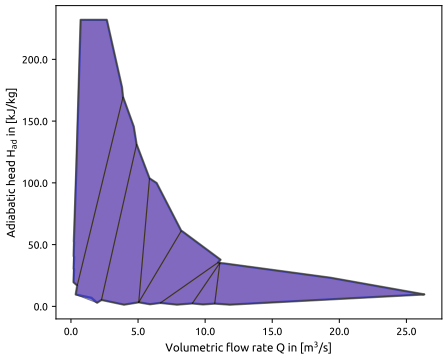
(d) 2 components, error = 15.55%.



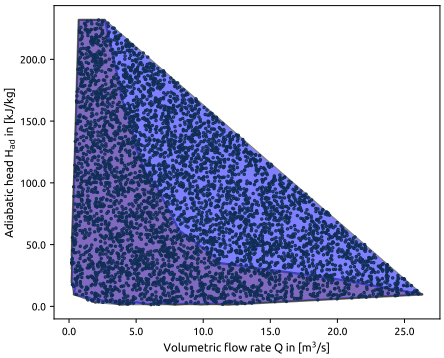
(e) 3 components, error = 6.71%.



(f) 4 components, error = 1.33%.



(g) 8 components, error = 0.25%.



(h) 5,000 sample instances.

Figure 5: Approximate convex decomposition of the aggregated characteristic diagram of an artificial compressor group. The compressor group consists of 6 compressor machines that can be operated in 42 different configurations.

# components	error(vol)	# feas	# infeas	total time
(42)	0.00%	2495	2505	987s
1	102.51%	5000	0	664s
2	15.55%	2842	2158	615s
3	6.71%	2676	2324	625s
4	1.33%	2530	2470	613s
5	0.75%	2514	2486	661s
8	0.25%	2503	2497	730s
13	0.00%	2498	2502	830s

Table 2: Computational results for ACD on operating range of compressor station.

model	# feas	time (feas)	# infeas	time (infeas)
physical	2287	5701s	2713	1694s
physical + ACD (1 comp.)	2294	5849s	2706	1686s
physical + ACD (2 comp.)	2292	5717s	2708	599s
physical + ACD (3 comp.)	2289	5770s	2711	495s
physical + ACD (4 comp.)	2296	6147s	2704	494s
physical + ACD (8 comp.)	2292	5873s	2708	782s
physical + ACD (13 comp.)	2291	6002s	2709	800s

Table 3: Computational results for ACD on operating range of compressor station.

the total solving time much. This is because the solver has to find solutions to the nonlinear compressor equations anyway. However, using approximate convex decompositions with two or more components reduces the running time for detecting infeasibility significantly. For those instances, the total runtime is reduced up to 70%.

5 Conclusions and outlook

In this paper we showed that the well-known "best possible" convex hull reformulation for disjunctive polyhedral sets may yield weak bounds in branch-and-bound search. This is due to the fact that this formulation does not take into account the *shape* of the entire set. We proposed to analyse this shape by computing a hierarchy of approximate convex decompositions, which can be used to augment the original formulation to provide additional branching opportunities. This yields considerable improvements in running time for solving the resulting models.

However, the algorithm from [14] that we used to compute our hierarchy has certain drawbacks for our application. First of all, the measure of nonconvexity used to determine where to refine the approximation is unrelated to the tightness of the resulting relaxation. Hence the resulting decomposition hierarchy

does not necessarily provide useful guidance for branching. For instance, in the first level of the decomposition hierarchy in Figure 4 the convex component corresponding to left child node is equal to the convex component of its parent. Moreover, nonconvexity is reduced in the classical way [6] by splitting the nonconvex polygon at a vertex that is not on the convex hull and another vertex (this is done by the Resolve-procedure). Due to this "local" cutting, the convex components of two sibling nodes in the hierarchy need not be relatively disjoint, which is however a desirable property for branch-and-bound search.

In fact, these drawbacks are shared by other existing methods for obtaining approximate convex decompositions [15, 1, 17, 16]. Therefore, there is need for a method that is tailored to this application.

Acknowledgements The authors thank the DFG for their support within project A04 in CRC TRR154 and the BMBF Research Campus Modal (fund number 05M14ZAM) and ICT COST Action TD1207 for additional support.

References

[1] Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. Hierarchical convex approximation of 3D shapes for fast region selection. *Computer Graphics Forum*, 27(5):1323–1333, 2008.

[2] Egon Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[3] Egon Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6:466–486, 1985.

[4] Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, 1998.

[5] Sebastián Ceria and João Soares. Convex programming for disjunctive convex optimization. *Math. Program., Ser. A*, 86:595–614, 1999.

[6] B. Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM J. Comput.*, 13(3):488–507, 1984.

[7] Armin Fügenschuh, Björn Geißler, Ralf Gollmer, Antonio Morsi, Marc E. Pfetsch, Jessica Rövekamp, Martin Schmidt, Klaus Spreckelsen, and Marc C. Steinbach. Physical and technical fundamentals of gas networks. In Koch et al. [12].

[8] Sean Gillies. Shapely python package. `https://github.com/Toblerity/Shapely`, 2008–2017.

[9] B. Hiller and T. Walther. Modelling compressor stations in gas networks. Technical report, Zuse Institute Berlin, 2017.

[10] Jesco Humpola, Armin Fügenschuh, Benjamin Hiller, Thorsten Koch, Thomas Lehmann, Ralf Lenz, Robert Schwarz, and Jonas Schweiger. The specialized MINLP approach. In Koch et al. [12].

[11] Jesco Humpola, Imke Joormann, Djamal Oucherif, Marc E. Pfetsch, Lars Schewe, Martin Schmidt, and Robert Schwarz. GasLib – A Library of Gas Network Instances. Technical report, Nov 2015.

[12] Thorsten Koch, Benjamin Hiller, Marc Pfetsch, and Lars Schewe, editors. *Evaluating Gas Network Capacities*. MOS-SIAM Series on Optimization. SIAM, 2015.

[13] Sangbum Lee and Ignacio E. Grossmann. New algorithms for nonlinear generalized disjunctive programming. *Computers and Chemical Engineering*, 24:2125–2141, 2000.

[14] Jyh-Ming Lien and Nancy M. Amato. Approximate Convex Decomposition of Polygons. *Computational Geometry*, 35(1–2):100–123, August 2006.

[15] Jyh-Ming Lien and Nancy M. Amato. Approximate Convex Decomposition of Polyhedra and Its Applications. *Computer Aided Geometric Design*, 25(7):503–522, October 2007.

[16] Guilin Liu, Zhonghua Xi, and Jyh-Ming Lien. Nearly convex segmentation of polyhedra through convex ridge separation. Technical Report GMU-CS-TR-2015-3, Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030-4444 USA, 2015.

[17] Hairong Liu, Wenyu Liu, and Longin Jan Latecki. Convex shape decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, 13-18 June 2010, San Francisco, CA, USA*, pages 97–104. IEEE, 2010.

[18] Stephen J. Maher, Tobias Fischer, Tristan Gally, Gerald Gamrath, Ambros Gleixner, Robert Lion Gottwald, Gregor Hendel, Thorsten Koch, Marco E. Lübbecke, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Dieter Weninger, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 4.0. ZIB Report 17–12, Zuse Institute Berlin, 2017.

[19] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[20] Juan P. Ruiz and Igancio E. Grossmann. A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. *European Journal of Operational Research*, 218:38–47, 2012.

[21] Roger Z. Ríos-Mercado and Conrado Borraz-Sánchez. Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, 147:536–555, 2015.

[22] Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.