Can we parameterize disjunctive cuts to improve solver performance for a sequence of MILPs? If so, how?

**THEORY**

**Input:** A sequence of mixed integer linear optimization problems (MILPs), \( \{P_1, \ldots, P_k\} \), sharing same variables.

\[
\min_{x \in \mathbb{R}^n} \quad c^T x \\
A x \geq b \\
x \in \mathbb{Z}
\]

Let \((X^t)_{\mathbb{R}^n}\) be a disjunction, where \(X^t := \{x \in \mathbb{R}^n ; D^t \geq D_0^t \} \). \((X^t)_{\mathbb{R}^n}\) is valid for a set \(S \subseteq \mathbb{R}^n\) if \(S \subseteq \bigcup_{t=1}^T X^t\). Let \(Q^T = P^T \cap X^t\).

Opportunity: If the sequence varies little, a MILP solver might employ similar disjunctions in solving each instance.

**Idea:** Generate \(\mathcal{V}\)-Polyhedral Disjunctive Cuts (VPCs) via \([1]\) for some instances and reapply them to the remaining instances.

**Problem:** VPCs can become invalid when constraints are perturbed.

**Solution:** After generating a VPC for \(P_1\), parameterize it to ensure its validity when applied to \(P_j\) for \(j \geq 2\).

**COMPUTATION**

**Experimental Setup:**
- The Base Set consists of 104 presolved MIPLIB 2017 instances with at most 5000 variables and 5000 constraints.
- The Experiment Set consists of 5 random perturbations of objective, RHS, and/or matrix for each instance in Base Set.
- Replications vary by the following parameters:
  - 4, 16, or 64-term disjunctions for VPC generation
  - 0.5 or 2 degrees of random perturbation
  - No VPCs, VPCs via \([1]\), or parameterized VPCs
- The Experiment Set is solved for each combination of parameters using Gurobi 10.

![Graphs showing parameterized VPC improvements](image)

<table>
<thead>
<tr>
<th>Degree</th>
<th>Terms</th>
<th>No VPCs</th>
<th>VPCs via [1]</th>
<th>Param. VPCs</th>
<th>No VPCs</th>
<th>VPCs via [1]</th>
<th>Param. VPCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>4</td>
<td>61.87%</td>
<td>62.13%</td>
<td>0.929</td>
<td>10.480</td>
<td>0.999</td>
<td>0.000%</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>61.87%</td>
<td>62.13%</td>
<td>0.929</td>
<td>10.480</td>
<td>0.999</td>
<td>0.000%</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>61.87%</td>
<td>62.13%</td>
<td>0.929</td>
<td>10.480</td>
<td>0.999</td>
<td>0.000%</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>61.46%</td>
<td>61.76%</td>
<td>0.870</td>
<td>17.756</td>
<td>1.394</td>
<td>0.558%</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>61.46%</td>
<td>61.76%</td>
<td>0.870</td>
<td>17.756</td>
<td>1.394</td>
<td>0.558%</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>61.46%</td>
<td>61.76%</td>
<td>0.870</td>
<td>17.756</td>
<td>1.394</td>
<td>0.558%</td>
</tr>
</tbody>
</table>

**Relative Improvements between Solves with and without Parameterized VPCs**

<table>
<thead>
<tr>
<th>Term</th>
<th>Nodes Processed</th>
<th>LP Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>33rd percentile</td>
<td>median</td>
<td>1.000</td>
</tr>
<tr>
<td>50th percentile</td>
<td>median</td>
<td>1.000</td>
</tr>
<tr>
<td>66th percentile</td>
<td>median</td>
<td>1.000</td>
</tr>
<tr>
<td>50th percentile</td>
<td>median</td>
<td>1.000</td>
</tr>
</tbody>
</table>

**CONCLUSION**

**Key Takeaways:**
- Parameterization amortizes the cost of generating VPCs via \([1]\), often still improving the strength of default cuts at the root.
- A significant number of perturbed instances see improvements to run time, nodes processed, and LP iterations.

**Next Steps:**
- Generalize \([1]\) to include infeasible disjunctive terms. Currently, \(v^t = 0\) for \(P_k\) generating VPCs via \([1]\) and \(t\) such that \(Q^{T=0}\). For \(\ell \geq k\) and \(Q^{T=0}\), this weakens parameterization occurring from Lemma 2.
- Better understand why parameterized VPCs help for some perturbations of the same degree and base instance but not others.

**References:**

Yes, we can! For \(P_k\), find disjunctive cuts and their Farkas multipliers. For \(P_\ell\) with \(\ell \geq k\), use Farkas multipliers to compute new valid inequalities.