A dynamic adaptive discretisation algorithm for the pooling problem

A Dynamic Adaptive Discretization Algorithm for the Pooling Problem

Akshay Gupte, Arie M.C.A. Koster, and Sascha Kuhnke

In this talk, we present an iterative algorithm based on discretization to calculate high quality solutions for the pooling problem in reasonable running time. The pooling problem was first introduced by Haverly (1978). This problem has important applications in industries such as refinery processes in the petroleum industry, wastewater treatment or gas transportation planning. In the pooling problem, a flow network $(\mathcal{N}, \mathcal{A})$ is given where the set of nodes $\mathcal{N}$ can be partitioned into a set of inputs I, a set of pools L, and a set of outputs J. Flow can be sent from inputs either directly or via pools to outputs, i.e., all arcs $\mathcal{A}$ are a subset of $\{(I \times L) \cup (I \times J) \cup (L \times J)\}$. Moreover, each node $n \in \mathcal{N}$ and each arc $a \in \mathcal{A}$ have capacities c_n and c_a, respectively. Each arc a also has a weight f_a for each unit of flow. Lastly, we have a set K of specifications with given concentrations λ_k^i at each input $i \in I$ as well as lower bounds $\underline{\lambda_k^j}$ and upper bounds $\overline{\lambda_k^j}$ at each output $j \in J$ for all $k \in K$. The goal is to maximize a linear function over the flow on each arc while keeping the specification values at each output between their lower and upper bounds.

Tawarmalani and Sahinidis (2002) proposed the *pq*-formulation of the pooling problem. This formulation contains flow variables y_{ij} for each arc $(i, j) \in \mathcal{A}$ with $i \in I \cup L$ and $j \in J$. Moreover, the variables v_{ilj} represent the flow that goes from input i via pool l to output j. For each pool l and input i, the variable q_{il} is the proportion of flow in arc (i, l) to the total inlet flow to pool l. The objective of the *pq*-formulation is to maximize a linear weight function over the flow variables:

$$\max \sum_{i \in I, j \in J} f_{ij}\, y_{ij} + \sum_{i \in I, l \in L, j \in J} (f_{il} + f_{lj})\, v_{ilj} \tag{1}$$

In the following, we give the constraints of the *pq*-formulation. The first two constraints define the proportion variables q_{il}:

$$v_{ilj} = q_{il}\, y_{lj} \qquad \forall\ i \in I, l \in L, j \in J \tag{2}$$

$$\sum_{i \in I} q_{il} = 1 \qquad \forall\ l \in L \tag{3}$$

The following two constraints ensure that the specification concentrations at each output lie between their lower and upper bounds:

$$\sum_{i \in I} \lambda_k^i\, y_{ij} + \sum_{i \in I, l \in L} \lambda_k^i\, v_{ilj} \geq \underline{\lambda_k^j} \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \qquad \forall\ k \in K, j \in J \tag{4}$$

$$\sum_{i \in I} \lambda_k^i\, y_{ij} + \sum_{i \in I, l \in L} \lambda_k^i\, v_{ilj} \leq \overline{\lambda_k^j} \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \qquad \forall\ k \in K, j \in J \tag{5}$$

Moreover, the flow and arc capacities have to be respected:

$$\sum_{j \in J} y_{ij} + \sum_{l \in L, j \in J} v_{ilj} \leq c_i \qquad \forall\, i \in I \tag{6}$$

$$\sum_{j \in J} y_{lj} \leq c_l \qquad \forall\, l \in L \tag{7}$$

$$\sum_{l \in L} y_{lj} \leq c_j \qquad \forall\, j \in J \tag{8}$$

$$\sum_{j \in J} v_{ilj} \leq c_{il} \qquad \forall\, i \in I, l \in L \tag{9}$$

$$y_{ij} \leq c_{ij} \qquad \forall\, i \in I \cup L, j \in J \tag{10}$$

To get a tighter formulation with stronger LP relaxations, we add two valid constraints which are already implied by the previous constraints:

$$\sum_{i \in I} v_{ilj} = y_{lj} \qquad \forall\, l \in L, j \in J \tag{11}$$

$$\sum_{j \in J} v_{ilj} \leq c_l\, q_{il} \qquad \forall\, i \in I, l \in L \tag{12}$$

Finally, all variables have to be non-negative:

$$y_{ij}, v_{ilj}, q_{il} \geq 0 \qquad \forall\, i \in I, l \in L, j \in J \tag{13}$$

Due to the bilinear constraints (2), the above formulation is a nonconvex quadratically constrained program (QCP) where nonlinear solvers have difficulties to find feasible solutions for real world instances.

We present an iterative algorithm to determine feasible solutions to the pooling problem. This algorithm is inspired by earlier, similar ideas presented for decentralized energy systems (Goderbauer et al., 2016) and waste water network design (Koster and Kuhnke, 2018). In each iteration, we solve a restriction of the original problem where we discretize both the number and the distribution of outlet flows at each pool. This restriction, first introduced by Dey and Gupte (2015), can be modeled as a mixed integer linear program (MILP) which is likely to be easier to solve than the original problem. To get more suitable discretizations, we iteratively adapt the discretization size along with the discretization values for each pool based on the previous MILP solution.

An extensive computational study on 70 medium- to large-scale test instances is conducted to show the effectiveness of this approach. Compared to nonlinear solvers as well as to other non-iterative discretization algorithms, our approach achieves the best results within short running times.

References

S.S. Dey and A. Gupte. Analysis of MILP Techniques for the Pooling Problem. *Operations Research*, 63(2):412–427, 2015.

S. Goderbauer, B. Bahl, P. Voll, M.E. Lübbecke, A. Bardow, and A.M.C.A. Koster. An adaptive discretization MINLP algorithm for optimal synthesis of decentralized energy supply systems. *Computers & Chemical Engineering*, 95:38–48, 2016.

C.A. Haverly. Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, (25):19–28, December 1978.

A.M.C.A. Koster and S. Kuhnke. An adaptive discretization algorithm for the design of water usage and treatment networks. *Optimization and Engineering*, Dec 2018.

M. Tawarmalani and N.V. Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications.* Kluwer Academic Publishers, 2002.