A Decomposition Heuristic for Mixed-Integer Supply Chain Problems

Lars Schewe, Martin Schmidt, Dieter Weninger

Abstract. Mixed-integer supply chain models typically are very large but are also very sparse and can be decomposed into loosely coupled blocks. In this paper, we use general-purpose techniques to obtain a block decomposition of supply chain instances and apply a tailored penalty alternating direction method, which exploits the structural properties of the decomposed instances. We further describe problem-specific enhancements of the algorithm and present numerical results on real-world instances that illustrate the applicability of the approach.

1. Introduction

Ongoing globalization and rapid changes in information technology produce significant challenges in managing operations in today's competitive market places. To address these challenges for integrated business planning, *Supply Chain Management (SCM)* has been introduced in 1982 in [16]. Since then, SCM attracts increasing attention both in industrial practice as well as in academic research. In [18], SCM is defined as the task of integrating organizational units along a supply chain and of coordinating materials, information, and financial flows in order to satisfy the customers' demands with the aim of improving competitiveness of the supply chain as a whole. One part of SCM is *Advanced Planning* and computer-based methods for this purpose are called Advanced Planning Systems (APS). APS are based on hierarchical planning [11] and make extensive use of instance-specific heuristics as well as of exact mathematical optimization. In the late 1990s, various software vendors started to offer APS that help companies to coordinate and plan their growing supply chains. Following this trend, SAP, as one of the leading vendors of business software, started to develop and sell products such as the *SAP Advanced Planning and Optimization* software. One specific function thereof is called Supply Network Planning Optimization (SNP optimization), which relies on mixed-integer linear optimization (MIP) models and solvers [6]. SNP optimization aims at providing quantitative decision support for typical supply chain processes such as material procurement, production, transportation, demand fulfillment, stock keeping, and resource capacity utilization. Some quantities may be produced and transported only in discrete lots. The granularity of these quantities is determined according to the planning horizon, which is subdivided into so-called time-buckets (or simply buckets). Typical supply chain scenarios possess a planning horizon of one or two years and may consist of up to about 100 time-buckets.

Solving real-world supply chain instances is rather hard since these models are typically of mixed-integer type and very large. Fortunately, these instances possess a special structure: They are very sparse and contain many so-called independent components [5]. These properties allow to decompose supply chain instances into

Date: February 18, 2020.

2010 *Mathematics Subject Classification.* 90Bxx, 90C11, 90C59, 90C90.

Key words and phrases. Supply chain, Mixed-integer optimization, Decomposition, Penalty method, Alternating direction method.

block-structured MIPs (cf. Section 2 and 3) in which the separate blocks are only coupled by comparably few variables; see, e.g., [3, 15] for general discussions of block-structured MIPs and [1, 14] for computational studies regarding block-detection and exploiting block-structure in MIPs. Moreover, see the very recent paper [19] for a combination of two classical MIP solution approaches; branch-and-bound and decomposition. In this paper, we present a tailored penalty alternating direction method (PADM) that exploits the block-structure of SNP instances and that serves as a theory-driven general-purpose primal heuristic. See [7] for a detailed description of PADM and [8, 9, 13], where it is successfully applied to gas network optimization and bilevel problems. We describe the general method, present problem-specific algorithmic enhancements in Section 4, and then discuss numerical results in Section 5. The presented method has the advantage that it can be used as a general-purpose heuristic that detects the block-structure of the problem automatically as well as a highly-tailored instance-specific heuristic if user knowledge about the structure of the instance is at hand. It turns out that the PADM heuristic is competitive with Gurobi w.r.t. finding feasible solutions of good quality quickly.

2. Problem Statement

For finding a feasible or even optimal solution for the supply chain problem, SNP optimization tools typically map the data into a MIP model. The variables of this model represent the planning decisions per time-bucket, the constraints constitute business-specific rules or restrictions, and the objective function represents the overall costs to be minimized. Some of the variables may be integral because supply chains often require certain quantities of production or transport to be planned in discrete lots. After SNP optimization has built the mathematical model, it typically instructs MIP software to solve it. The solver's result is then converted into a supply chain plan proposed to the planner or customer.

The supply chain structure itself yields numerous different constraints. These emerge from restrictions entered by the customer like limited resource capacities but also from essential requirements as stock balance consistency. In what follows, we present a basic model containing typical elements of more general or complex supply chain problems. Our aim here is to keep the setup rather simple to later focus on the algorithmic development exploiting the structure of the problem at hand.

The presented model is based on the sets, parameters, and variables defined in Table 1 and 2. All parameters of the model are denoted with Greek letters whereas all variables are denoted with Latin letters and all variables are given a default lower bound of 0. Upper bounds are implied by the constraints. Apart from the binary indicator variables $w_o(t)$ for minimum lot sizes, all variables of the model are continuous. Not that in practical applications, some other variables may also be restricted to discrete values. For instance, the variables $u_o(t)$ are often required to be discrete, which allows to realize fixed lot sizes for production. Also, it is quite common to use discrete transport variables $z_{ap}(t)$.

The main goal of SCM is to satisfy requested demands. However, there are no cost-relevant benefits or rewards generated by satisfying a demand. Instead, in order to initiate activity within the supply chain—besides the costs of production and transportation—the model incorporates non-delivery penalties. A demand d for product p may allow a late delivery by at most $\delta_{dp}(t)$ time-buckets. To obtain the non-delivered amount regarding a demand in bucket t, all deliveries for that demand, even the delayed ones, are subtracted from the original demand quantity

TABLE 1. Basic sets used in the supply chain model

Symbol	Explanation
T	Time-buckets; $T = \{1, \ldots, \lvert T\rvert\}$
L	Locations; nodes of the transport graph $G = (L, A)$
A	Arcs of the transport graph G
$\delta^{\text{in}}(l)$	Transport arcs with location $l \in L$ as destination
$\delta^{\text{out}}(l)$	Transport arcs with location $l \in L$ as origin
D	Demands
D_t	Demands in bucket $t \in T$
D_l	Demands at location $l \in L$
P	Products
P_l	Products handled at location $l \in L$
P_a	Products transported on arc $a \in A$
O_l	Production models at location $l \in L$
R_l	Resources at location $l \in L$

TABLE 2. Parameters (above) and variables (below) of the model

Symbol	Explanation
$\alpha_{dp}(t)$	Quantity of demand d for product p in bucket t
$\delta_{dp}(t)$	Maximum allowed lateness for demand d for product p in bucket t
$\beta_{dp}(t)$	Non-delivery costs for demand d for product p in bucket t
$\gamma_{dp}(t, t')$	Late-delivery costs for delivering demand d from bucket t for product p if it is $t' - t$ buckets late
$\zeta_{ap}(t)$	Costs for transporting one unit of product p via arc a in bucket t
$\eta_o(t)$	Costs of applying production model $o \in O_l$ once at location l in bucket t
$\theta_{lp}(t)$	Costs of procuring product p at location l in bucket t
$\pi^+_{op}(t)$	Output quantity of product p from production model o in bucket t
$\pi^-_{op}(t)$	Input quantity of product p to production model o in bucket t
$\sigma_{lp}(t)$	Costs of storing product p at location l in bucket t
$\rho_{or}(t)$	Requirement of resource $r \in R_l$ for production model $o \in O_l$ at location l and in bucket t
$\psi_r(t)$	Capacity of resource $r \in R_l$ at location l and bucket t
$\varphi_o(t)$	Minimum lot size for production model $o \in O_l$ at location l and in bucket t
M	Large constant used in big-M method
$x_{dp}(t)$	Quantity not delivered for demand d of product p in bucket t
$y_{dp}(t, t')$	Quantity delivered in bucket t' for demand d of product p in bucket t
$z_{ap}(t)$	Quantity of product p transported on arc a in bucket t
$u_o(t)$	Number of applications of production model $o \in O_l$ at location l in bucket t
$v_{lp}(t)$	Quantity procured of product p at location l in bucket t
$s_{lp}(t)$	Stock level at location l of product p in bucket t
$w_o(t)$	Binary indicator variable for minimum lot sizes

in that bucket:

$$x_{dp}(t) = \alpha_{dp}(t) - \sum_{t'=t}^{t+\delta_{dp}(t)} y_{dp}(t, t') \quad \text{for all } d \in D,\ p \in P,\ t \in T. \tag{1}$$

The following stock-level balance equation contains all information on input and output to and from a location by production, transport, or procurement as well as on stock quantities from the previous time-bucket. For the initial stock level $s_{lp}(0)$, usually a value of zero is assumed. Note that in the stock-level balance equation for bucket $t \in T$ possible late deliveries to satisfy demands of product p from earlier time-buckets $\tau \in T$ need to be taken into account as long as their maximum allowed lateness $\delta_{dp}(\tau)$ permits a delivery up to time-bucket t, i.e., $t - \delta_{dp}(t) \leq \tau \leq t$:

$$\begin{aligned} s_{lp}(t) = s_{lp}(t-1) + v_{lp}(t) + \sum_{o \in O_l} \pi^{+}_{op}(t) u_o(t) + \sum_{a \in \delta^{\text{in}}(l)} z_{ap}(t) \\ - \sum_{o \in O_l} \pi^{-}_{op}(t) u_o(t) - \sum_{a \in \delta^{\text{out}}(l)} z_{ap}(t) - \sum_{d \in D_l} \sum_{\tau \leq t} y_{dp}(\tau, t) \\ \text{for all } l \in L,\ p \in P_l,\ \tau, t \in T. \end{aligned} \tag{2}$$

Resource restrictions can be considered for different activities. For simplicity, we only refer to production as a showcase. Resource capacities in production usually represent allocatable time on production machines or available man hours. Usually, different product lines may share the same resources and there are no costs for resource consumption. Production resource capacity restrictions are realized by

$$\sum_{o \in O_l} \rho_{or}(t) u_o(t) \leq \psi_r(t) \quad \text{for all } l \in L,\ r \in R_l,\ t \in T. \tag{3}$$

The minimum lot size for production describes the minimum number of applications of a production model. Two constraints are needed to model minimum lot sizes. First, we need to decide whether production takes place or not (which is modeled with the big-M method) and, second, we need to model the minimum quantity requirement:

$$u_o(t) - M w_o(t) \leq 0 \quad \text{for all } l \in L,\ o \in O_l,\ t \in T, \tag{4a}$$

$$\varphi_o(t) w_o(t) - u_o(t) \leq 0 \quad \text{for all } l \in L,\ o \in O_l,\ t \in T. \tag{4b}$$

Finally, the objective function is obtained by summing up all costs that are to be minimized. Thus, the entire problem is given by

$$\min_{\substack{x,y,z,s,\\ u,v,w}} \quad \sum_{t \in T} \Bigg(\sum_{d \in D_t} \sum_{p \in P} \sum_{t' > t}^{t + \delta_{dp}(t)} \Big(\gamma_{dp}(t,t') y_{dp}(t,t') + \beta_{dp}(t) x_{dp}(t) \Big) \tag{5a}$$

$$+ \sum_{a \in A} \sum_{p \in P_a} \zeta_{ap}(t) z_{ap}(t) + \sum_{l \in L} \sum_{o \in O_l} \eta_o(t) u_o(t) \tag{5b}$$

$$+ \sum_{l \in L} \sum_{p \in P_l} \Big(\theta_{lp}(t) v_{lp}(t) + \sigma_{lp}(t) s_{lp}(t) \Big) \Bigg) \tag{5c}$$

$$\text{s.t.} \quad (1)\text{–}(4). \tag{5d}$$

It should be noted that, in contrast to representations of production planning models in [17], the binary indicator variables $w_o(t)$ do not appear in the objective function coefficients of zero and thus contribute only implicitly over $u_o(t)$ to the value of the objective function. However, this aspect has no effect on the later proposed method for solving large-scale SNP instances.

The variables and constraints described above comprise a basic core of typical SNP models. In addition, there are many further features that we omit here in order to focus on the core model ingredients. These further features include, e.g.,

fixed lot sizes: the requirement to produce only multiples of a fixed quantity,
safety stock: keep material inventory above a certain minimum level,

TABLE 3. SNP Optimization instances

ID	cons	vars	intvars	nnz	0-vars	primal	dual	gap	time
scm01	65 473	111 628	4738	231 430	84 868	4.90×10^{8}	4.45×10^{8}	9.07	18 000
scm02	189 286	330 378	9522	681 772	249 924	6.08×10^{8}	5.74×10^{8}	5.48	18 001
scm03	98 871	169 018	6918	350 124	128 235	5.69×10^{8}	5.05×10^{8}	11.25	18 000
scm04	129 662	221 437	9092	459 205	168 562	7.85×10^{8}	7.33×10^{8}	6.55	18 000
scm05	257 078	441 219	18 252	916 248	333 129	1.00×10^{9}	9.49×10^{8}	5.42	18 001
scm06	195 616	334 684	13 436	693 612	305 026	1.13×10^{10}	1.83×10^{9}	83.78	18 000
scm07	320 836	549 020	22 118	1 138 755	517 166	2.59×10^{10}	5.77×10^{9}	77.68	18 000
scm08	648 150	1 115 101	44 398	2 313 528	1 050 307	5.04×10^{10}	9.83×10^{9}	80.49	18 001
scm09	1 388 032	2 589 531	1265	10 289 151	2 469 885	4.44×10^{11}	4.44×10^{11}	0.00	107
scm10	2 911 965	5 423 599	9694	20 504 298	5 133 363	5.76×10^{12}	5.76×10^{12}	0.00	413
scm11	26 360	84 578	189	239 740	61 012	5.86×10^{7}	5.83×10^{7}	0.39	18 000
scm12	29 407	94 955	210	309 292	68 930	7.09×10^{8}	3.37×10^{8}	52.54	18 000
scm13	28 169	92 212	231	286 368	68 195	1.79×10^{8}	1.78×10^{8}	0.49	18 000
scm14	121 006	371 681	1050	1 227 952	264 274	1.56×10^{8}	1.37×10^{8}	12.20	18 001
scm15	229 244	724 777	1750	2 193 682	529 058	1.34×10^{9}	6.14×10^{8}	54.21	18 000
scm16	131 742	424 074	2014	1 312 760	311 283	1.84×10^{9}	1.83×10^{9}	0.70	18 001
scm17	23 110	44 591	5337	107 496	33 453	8.98×10^{12}	8.98×10^{12}	0.00	417
scm18	83 387	382 995	3676	751 995	277 966	2.32×10^{16}	2.32×10^{16}	0.00	7105
scm19	226 605	227 316	91 462	1 062 673	216 549	3.17×10^{12}	3.17×10^{12}	0.27	18 000
scm20	80 852	174 824	8887	354 806	112 938	1.19×10^{14}	1.19×10^{14}	0.07	18 000
scm21	40 583	67 843	20 886	230 225	56 740	8.71×10^{8}	8.71×10^{8}	0.02	18 000
scm22	53 584	87 991	26 210	309 421	73 365	1.68×10^{9}	1.68×10^{9}	0.02	18 000
scm23	73 084	118 266	34 196	425 993	98 557	2.60×10^{9}	2.60×10^{9}	0.02	18 000
scm24	19 112	40 797	2170	97 333	26 791	-2.93×10^{8}	-2.94×10^{8}	0.11	18 000
scm25	21 663	46 207	2448	110 118	30 532	-3.31×10^{8}	-3.31×10^{8}	0.11	18 000
scm26	38 521	83 403	4433	198 419	55 367	-5.82×10^{8}	-5.82×10^{8}	0.12	18 000
scm27	50 773	137 703	6787	302 871	105 950	5.20×10^{14}	5.20×10^{14}	0.00	20
scm28	73 053	207 352	7431	459 840	161 100	5.20×10^{14}	5.20×10^{14}	0.00	76
scm29	108 710	318 396	7431	710 150	245 990	6.79×10^{14}	6.79×10^{14}	0.00	1135
scm30	87 260	383 794	1443	1 593 694	334 291	2.45×10^{11}	2.45×10^{11}	0.00	47
scm31	150 529	612 007	4633	2 512 992	539 938	2.45×10^{11}	2.45×10^{11}	0.00	536
scm32	29 264	61 966	3214	331 730	36 573	1.52×10^{11}	1.52×10^{11}	0.00	936
scm33	98 357	181 483	7866	1 560 108	112 203	1.75×10^{11}	1.62×10^{11}	7.39	18 000
scm34	189 919	354 328	8563	2 921 421	222 337	2.01×10^{11}	1.79×10^{11}	11.18	18 006
scm35	271 263	288 070	103 070	866 238	277 371	6.16×10^{11}	6.16×10^{11}	0.00	96
scm36	84 372	129 580	34 196	449 062	99 956	2.63×10^{9}	2.63×10^{9}	0.02	18 000
scm37	51 480	87 651	3573	206 186	49 191	-5.81×10^{8}	-5.82×10^{8}	0.11	18 000
scm38	113 566	323 072	7431	721 168	247 824	5.98×10^{15}	5.98×10^{15}	0.00	298
scm39	169 291	634 241	6535	2 433 653	557 851	2.47×10^{11}	2.47×10^{11}	0.00	372
scm40	265 554	438 373	9638	3 097 682	279 949	2.68×10^{11}	2.40×10^{11}	10.39	18 001

shelf life: limit material inventory not to exceed a certain maximum level,
capacity extension: extend resource capacity at some costs, or
cost functions: convex and concave piecewise linear cost functions on basic decisions such as production, transport, or inventory.

Let us finally note that, in principle, our heuristic can be applied to all these variants of SNP models. However, our method prefers models that can be decoupled into blocks so that only a few linking variables remain. This will be discussed in the next section.

3. Sparsity and Block-Structure of SNP Instances

The basis of our experiments is a set of 60 real-world supply chain instances from our industry partner SAP. SNP instances typically contain many independent components [5], i.e., parts of the original problem that share no common variables and constraints. The 60 instances contain more than 14 000 independent components. We have chosen all components with more than 100 integer variables and for which Gurobi needs at least 20 s for solving them to optimality. In addition, for composing a representative selection of instances, we preferred components that have different original instances as sources. This leads to a test set of 40 SNP instances, cf. Table 3, that all consist of a single component.

The computational results in Table 3 have been obtained using Gurobi 8.0.1 [10] (with default parameter settings and a time limit of 18 000 s) on a HPC cluster with

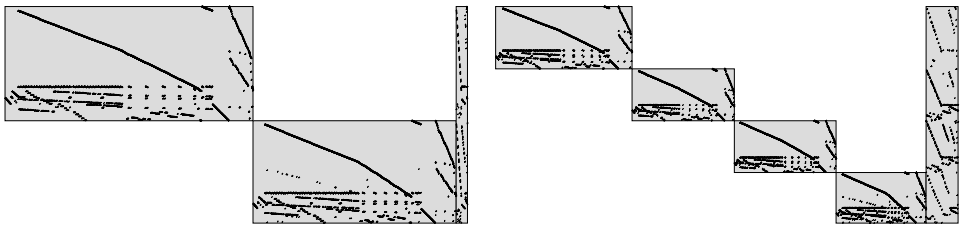

FIGURE 1. Decomposition of the instance scm24 into 2 (left) and 4 blocks (right).

Intel Xeon E5-2680 v4 ("Broadwell") processors 2.4 GHz and 512 GB RAM. The columns specify the name of the instance (ID), the number of constraints (cons), the number of variables (vars), the number of integer variables (intvars), the number of non-zeros (nnz) in the constraint matrix, the number of variables with value zero in the best solution (0-vars), the primal (b_p) and dual bound (b_d), the corresponding gap (gap), and finally the time for determining the best solution (time; in s). Gurobi was able to solve 13 instances to optimality within the time limit. However, the test set also contains very hard instances for Gurobi, e.g., scm06–scm08, scm12, or scm15.

From Table 3 it becomes obvious that SNP instances have two typical properties: (i) They are very sparse and (ii) the majority of the variables are 0 in a solution. Property (i) together with the existence of many loosely coupled blocks motivates us to apply a hypergraph partitioning approach to these instances. Our aim is to decompose the instances into blocks so that we obtain a minimum number of linking variables, i.e., of variables that connect different blocks of the decomposed instance. Property (ii) will turn out to be a valuable observation that we later exploit in our tailored decomposition method.

For decomposing the SNP instances we use k-way hypergraph partitioning as it is implemented in the software package hmetis; cf. [12] for the details. For hmetis, we use the imbalance tolerance 5. Figure 1 (left) shows a decomposition of the instance scm24 into two blocks (highlighted in gray) with 1004 linking variables, which form the full columns block at the right end of the matrix. The black dots indicate non-zero entries of the matrix. The same instance decomposed into four blocks is shown in the right plot of the figure. The number of linking variables is 2826. We can observe that a decomposition into more blocks usually results in a larger number of linking variables.

4. ALGORITHM

The basic problem (5) described in the last section has a so-called quasi block-separable structure. That means, it decomposes rather naturally in subproblems that are only weakly coupled. For such problems, the class of penalty alternating direction methods (PADM) have proven useful for practical applications; see [7] for a detailed description and theoretical analysis of these methods as well as [8, 9, 13] for applications of PADMs. Here, we briefly review these methods in Section 4.1, show how to apply the method to block-structured SNP instances in Section 4.2, and present some problem-specific algorithmic enhancements in Section 4.3.

4.1. **Basic Framework.** In a nutshell, a penalty alternating direction method is a standard alternating direction method (ADM) that is embedded within a penalty framework. For describing this class of methods let us consider optimization

Algorithm 1 The ℓ_1 Penalty Alternating Direction Method

Choose initial values $(x^{0,0}, y^{0,0}) \in X \times Y$ and penalty parameters $\mu^0 \geq 0$.
for $j = 0, 1, \ldots$ **do**
  Set $i = 0$.
  while $(x^{j,i}, y^{j,i})$ is not a partial minimum of (7) with $\mu = \mu^j$ **do**
    Compute $x^{j,i+1} \in \arg\min_x \{\phi_1(x, y^{j,i}; \mu^j) \colon x \in X\}$.
    Compute $y^{j,i+1} \in \arg\min_y \{\phi_1(x^{j,i+1}, y; \mu^j) \colon y \in Y\}$.
    Set $i \leftarrow i + 1$.
  end while
  Choose new penalty parameters $\mu^{j+1} \geq \mu^j$.
end for

problems of the form

$$\min_{x,y} \quad f(x,y) \quad \text{s.t.} \quad x \in X,\ y \in Y,\ g(x,y) = 0. \tag{6}$$

Here and in what follows, we assume that the objective function f and the so-called coupling constraints g are continuous and that the disjoint feasible sets $X \subseteq \mathbb{R}^{n_x} \times \mathbb{Z}^{m_x}$ and $Y \subseteq \mathbb{R}^{n_y} \times \mathbb{Z}^{m_y}$ are non-empty and compact. Note that we allow both the variable vector x and y to be mixed-integer. We also make the informal assumption that the number k of coupling constraints $g = (g_i)_{i=1}^k$ is small. Obviously, without this weak coupling the feasible set of the problem completely decomposes into the disjoint sets X and Y.

Classical ADMs proceed as follows. Given an iterate (x^j, y^j) they fix y to y^j and solve Problem (6) for x only. The result is the new x-iterate x^{j+1}. Then, x is fixed to x^{j+1} and Problem (6) is solved for y only, giving the new y-iterate y^{j+1}. ADMs repeat this procedure until there is no more progress, i.e., until $(x^j, y^j) = (x^{j-1}, y^{j-1})$ holds for some j. Such points are called partial minima (see, e.g., [7]) and it can be shown that ADMs converge to partial minima under rather mild assumptions.

In practice, it turns out that the coupling constraints g may significantly harm the solution process of the ADM. This motivates to remove these conditions from the constraint set in (6) and instead penalizing them in a weighted ℓ_1 penalty term:

$$\min_{x \in X, y \in Y} \quad \phi_1(x, y; \mu) := f(x,y) + \sum_{i=1}^{k} \mu_i |g_i(x,y)|. \tag{7}$$

A penalty ADM (PADM) now proceeds very similar to standard ADMs. For a given iterate we solve the two separate subproblems for (7) in x- and y-direction until no more progress is made. In this case, it is checked whether the coupling constraints are satisfied, i.e., if $\|g(x,y)\|_1 = 0$ holds. If this is the case, the PADM terminates; if not, the penalty parameters μ_i are updated. The formal listing for this method is given in Algorithm 1. For a more thorough discussion and the respective convergence theory, we refer the interested reader to [7].

4.2. **PADM Applied to Block-Decomposed SNP Instances.** We now briefly describe how to apply the PADM to the block-decomposed instances described in Section 2. For the ease of presentation, we explain the main ideas for the case of a decomposed problem with two blocks like in Figure 1 (left). That is, we have two blocks of variables and constraints $x \in X$ and $y \in Y$ as well as linking variables (with indices in $\mathcal{L}$) that appear in both blocks. Every such linking variable is now copied, yielding two new variables x_i and y_i that replace the original linking variable. The variable x_i is supposed to belong to X and y_i is supposed to belong to Y. Since these copies need to have the same value, all coupling constraints in g are of the

type $x_i = y_i$. Hence, Problem (7) is of the form

$$\min_{x \in X, y \in Y} \quad \phi_1(x, y; \mu) := f(x, y) + \sum_{i \in \mathcal{L}} \mu_i |x_i - y_i|.$$

The absolute values can be reformulated and the first PADM subproblem (i.e., for y fixed to $\bar{y}$) then reads

$$\begin{aligned} \min_{x \in X, s} \quad & \phi_1(x, s; \mu) := f(x, \bar{y}) + \sum_{i \in \mathcal{L}} \left(\mu_i^+ s_i^+ + \mu_i^- s_i^- \right) \\ \text{s.t.} \quad & s_i^+ - s_i^- = x_i - \bar{y}_i, \ s_i^+, s_i^- \geq 0 \quad \text{for all } i \in \mathcal{L} \end{aligned}$$

with $s^+ = (s_i^+)_{i \in \mathcal{L}}$, $s^- = (s_i^-)_{i \in \mathcal{L}}$, and $s = ((s^+)^\top, (s^-)^\top)^\top$. The same can be done for the second PADM subproblem in an analogous way. Note that applying this decomposition and the respective PADM ensures convergence to partial minima of the original problem; cf., e.g., [7].

4.3. **Problem-Specific Enhancements.** In the plain version as described in the last section, it turns out that the method is rather sensitive w.r.t. algorithmic parameters like the MIP gap used for solving the subproblems or w.r.t. how we update the penalty parameters. To this end, we next present some enhancements and implementation details of the general algorithm described in Section 4.1.

4.3.1. *Sigmoid Rescaling of Penalty Parameters.* Increasing the penalty parameters $\mu \in \mathbb{R}^k$ needs to be performed carefully since too large penalty parameters in the objective function may lead to numerical instabilities. To this end, we apply a sigmoid rescaling of the penalty parameters in certain cases. In what follows, let $S : \mathbb{R} \to \mathbb{R}$ be a sigmoid function having an "S"-shaped graph like the simple sigmoid function given by $S(x) := x/(1 + |x|)$.

Let $c \in \mathbb{R}^n$ be the coefficients of the original objective function. If $\|\mu\|_\infty \geq \theta \|c\|_\infty$ for some $\theta > 1$, we apply the sigmoid-based penalty parameter rescaling

$$\bar{S}(\mu_i) := \beta \frac{\mu_i - \sigma}{\alpha + |\mu_i - \sigma|} + \omega \quad \text{for all } i = 1, \ldots, k.$$

The parameters σ and ω lead to shifts of the sigmoid function and $\beta \geq 1$ controls the width of the rescaling interval. Finally, the parameter $\alpha \geq 1$ controls the flatness of the curve. By applying $\bar{S}$, all penalty parameters are mapped into a controllable interval and since $\bar{S}$ is a monotonically increasing function, the ordering of the penalty parameters w.r.t. their size stays the same and the medium-sized penalty parameters are compressed around the center of the interval. Based on our extensive preliminary numerical experiments, we chose the parameter values $\alpha = \|\mu\|_\infty$, $\sigma = \|\mu\|_\infty$, $\beta = 5$, $\omega = 5$, and $\theta = 10^3$.

4.3.2. *Influence of the MIPGap Parameter.* The MIP solver Gurobi allows to adjust the parameter MIPGap, which has a default value of 10^{-4}. Let b_p be the bound of the best known incumbent solution and let b_d be the current dual bound. If

$$|b_p - b_d| \leq \mathsf{MIPGap} \cdot |b_p|$$

is satisfied, Gurobi terminates with the current incumbent solution. Using larger values for the MIPGap parameter leads to Gurobi stopping earlier with an approximate solution. Our preliminary numerical experiments revealed that in most cases the quality of subproblem solutions caused by larger MIPGap values are sufficient for the algorithm to converge. The respective trade-off is as expected: We observe that using larger values often results in a considerably shorter running times of our algorithm. On the other hand, applying smaller values sometimes leads to PADM solutions of better quality.

In our implementation of Algorithm 1 we use an adaptive rule for choosing the MIPGap parameter. We start with a large value in order to obtain solutions of the early subproblems quickly and then reduce this value to $\max\{\mathsf{MIPGap}/2, 10^{-3}\}$ in two situations: First, if the inner ADM loop of the algorithm does not converge and, second, after every sigmoid rescaling of the penalty parameters. We use the same MIPGap value for all subproblems.

4.3.3. *Handling of Unbounded Subproblems.* Decomposing a mixed-integer linear program may lead to unbounded subproblems. Consider the following mixed-integer linear program

$$\min_{x_1,x_2,x_3} \quad -2x_1 \tag{8a}$$

$$\text{s.t.} \quad x_1 - x_2 \leq 0, \quad x_2 + x_3 = 3, \quad x_1, x_2, x_3 \geq 0, \quad x_1 \in \mathbb{Z}, \tag{8b}$$

with optimal solution $(x_1^*, x_2^*, x_3^*) = (3, 3, 0)$, which is also the solution of the linear programming relaxation. In this example, the only linking variable is x_2. Thus, decomposing Problem (8) as described in Section 4.2 yields the first subproblem

$$\min_{x_1,x_2',s_2^+,s_2^-} \quad -2x_1 + s_2^+ + s_2^- \tag{9a}$$

$$\text{s.t.} \quad x_1 - x_2' \leq 0, \quad s_2^+ - s_2^- = x_2' - \bar{x}_2'', \quad x_1, x_2', s_2^+, s_2^- \geq 0. \tag{9b}$$

The second subproblem can be stated in an analogous way. In the first iteration, the initial value of $\bar{x}_2''$ could be chosen, e.g., as the corresponding value (3) in the LP relaxation.

In our implementation, we initialize the penalty parameters of the slack variables using the objective function coefficient of the corresponding variables incremented by 1. In the example, this leads to a penalty parameter of 1 for s_2^+ and s_2^-. In this case, however, it can be seen that Subproblem (9) is unbounded although the original problem (8) has an optimal solution. This issue cannot appear for sufficiently large penalty parameters. In our implementation, we solve the current subproblem and check if it is unbounded. If this is the case, we iteratively increase all penalty parameters by a factor of 10 until the subproblem gets bounded. Note that, in the above example, it is sufficient to set the penalty parameters to 2 to achieve a bounded subproblem.

5. Numerical Results

The numerical results presented in this section have been obtained using HPC cluster machines with Intel Xeon E5-2643 v4 ("Broadwell") processors with 3.4 GHz and 256 GB RAM, running Linux in 64 bit mode. We have implemented our algorithm using the C++ API of Gurobi 8.0.1. Within our algorithm, Gurobi is always applied with its default settings except of the initial value $\mathsf{MIPGap} = 2$ for solving the individual subproblems within the PADM. All penalty parameters are updated using a factor of 10.

Since, to the best of our knowledge, there are no general-purpose supply chain heuristics to compare with (see, e.g., [6]), we compare different parameter settings of our algorithm with Gurobi running with default parameter settings except for the parameter MIPFocus, which is set to 1, so that Gurobi is focusing on finding feasible points quickly. To this end, we apply the PADM until a first feasible solution is found and use the time required by PADM as the time limit for Gurobi. We then analyze whether Gurobi is also able to compute a feasible solution within this time limit and, if this is the case, how the quality of the feasible solutions compare to each other. For brevity, we only present instance decompositions into 2 or 4 blocks because using more than 4 blocks always increased the number of linking variables and typically lead to larger running times in our preliminary numerical experiments.

TABLE 4. Numerical results for 2 blocks, zero objective function coefficients, and 0 as initial values for the linking variables. All times are given in seconds.

	Gurobi 8.0.1		PADM			
ID	obj. value	time	obj. value	$\|\mathcal{L}\|$	h-time	time
scm01	4.03×10^{9}	4	4.03×10^{9}	514	3	4
scm02	1.17×10^{10}	14	$\mathbf{8.28 \times 10^{9}}$	851	11	14
scm03	6.58×10^{9}	7	6.58×10^{9}	771	5	7
scm04	8.60×10^{9}	9	8.60×10^{9}	1028	7	9
scm05	1.64×10^{10}	20	1.64×10^{10}	2026	16	19
scm06	1.38×10^{10}	15	1.38×10^{10}	1541	12	15
scm07	2.59×10^{10}	27	2.59×10^{10}	2568	21	27
scm08	5.04×10^{10}	61	5.04×10^{10}	4998	54	61
scm09	$\mathbf{4.44 \times 10^{11}}$	95	4.61×10^{11}	601	199	223
scm10	$\mathbf{5.76 \times 10^{12}}$	353	1.32×10^{13}	877	505	556
scm11	–	3	$\mathbf{2.66 \times 10^{9}}$	84	2	3
scm12	3.44×10^{11}	4	$\mathbf{9.22 \times 10^{9}}$	105	3	4
scm13	$\mathbf{1.17 \times 10^{9}}$	4	1.34×10^{9}	105	2	4
scm14	2.05×10^{12}	18	$\mathbf{2.42 \times 10^{9}}$	336	15	18
scm15	–	34	$\mathbf{8.60 \times 10^{9}}$	578	29	34
scm16	–	19	$\mathbf{8.73 \times 10^{9}}$	42	16	19
scm17	1.59×10^{16}	2	$\mathbf{6.38 \times 10^{15}}$	835	1	2
scm18	–	22	$\mathbf{9.56 \times 10^{16}}$	2640	14	22
scm19	$\mathbf{3.23 \times 10^{12}}$	39	2.26×10^{13}	2360	29	39
scm20	–	7	$\mathbf{5.61 \times 10^{14}}$	688	5	7
scm21	$\mathbf{8.84 \times 10^{8}}$	11	9.37×10^{9}	1751	3	11
scm22	$\mathbf{1.74 \times 10^{9}}$	6	1.21×10^{10}	2184	5	6
scm23	$\mathbf{2.67 \times 10^{9}}$	11	1.66×10^{10}	2850	9	11
scm24	$\mathbf{-2.91 \times 10^{8}}$	3	-2.45×10^{8}	1004	2	3
scm25	$\mathbf{-3.28 \times 10^{8}}$	3	-2.74×10^{8}	762	2	3
scm26	$\mathbf{-5.75 \times 10^{8}}$	6	-4.83×10^{8}	734	4	6
scm27	$\mathbf{5.21 \times 10^{14}}$	6	1.20×10^{18}	1159	4	6
scm28	$\mathbf{5.21 \times 10^{14}}$	9	2.96×10^{18}	2072	7	9
scm29	$\mathbf{6.80 \times 10^{14}}$	16	2.90×10^{18}	3472	11	16
scm30	$\mathbf{2.45 \times 10^{11}}$	14	3.35×10^{11}	55	17	20
scm31	–	36	$\mathbf{7.12 \times 10^{11}}$	363	30	36
scm32	$\mathbf{5.89 \times 10^{11}}$	4	1.55×10^{12}	115	3	4
scm33	–	17	$\mathbf{1.26 \times 10^{13}}$	629	13	17
scm34	–	45	$\mathbf{2.56 \times 10^{13}}$	1644	33	45
scm35	$\mathbf{6.25 \times 10^{11}}$	22	2.74×10^{12}	332	17	22
scm36	$\mathbf{2.70 \times 10^{9}}$	13	1.75×10^{10}	2853	8	13
scm37	$\mathbf{-5.76 \times 10^{8}}$	5	-5.12×10^{8}	464	3	5
scm38	$\mathbf{5.98 \times 10^{15}}$	17	9.02×10^{17}	3478	13	17
scm39	–	36	$\mathbf{7.29 \times 10^{11}}$	441	31	36
scm40	–	45	$\mathbf{2.56 \times 10^{13}}$	1632	39	45

For initializing the linking variables we tested two methods: (i) Initial values of 0 or (ii) using the corresponding value of the LP relaxation. The first method is motivated by Property (ii) in Section 3. Moreover, the handling of the original objective function has a major impact on solving the PADM subproblems. Again, we have tested two different versions. First, we use the objective function with its original coefficients. Second, we ignore the original objective function, i.e., we set all respective coefficients to 0. The results are as expected: Using the original objective function can lead to feasible solutions with better objective function value but typically increases the running time significantly. Since our main focus is on finding feasible solutions quickly, we only present results for the case in which we ignore the original objective function. Note that this is comparable to the rationale in early publications on the feasibility pump heuristic [2, 4].

We discuss the results for using 2 blocks, setting the objective function to 0, and using initial values of zero for the linking variables in detail; see Table 4. The results for the case of 2 blocks and using the LP relaxation as initial values for the linking variables as well as the case of 4 blocks and using zero as initial values for the linking variables are given in Table 5 and 6 in the appendix. The first column in Table 4 denotes the ID of the instance, the second column contains the objective value obtained by Gurobi, the third column displays the running time of Gurobi, the

fourth column illustrates the objective function value obtained with the PADM, the fifth column contains the number of linking variables ($|\mathcal{L}|$), the sixth column (h-time) contains the time used by hmetis, and the last column states the complete time used by PADM (including the hmetis time).

First, it can be seen that the PADM finds a feasible solution for every SNP instance. A dash indicates that Gurobi was not able to find a solution within the time required by the PADM. This was the case for 10 instances, i.e., for 25 % of all instances PADM finds a feasible solution faster than Gurobi. In 19 cases Gurobi determines a better solution and in 14 cases PADM determines a better solution. The latter result is quite surprising since we completely ignore the original objective function. The better solution is always printed in bold in the table. For 7 instances Gurobi and PADM achieved the same solution. Taking a closer look into Table 5 and 6 in the appendix, one can see that the general behavior stays the same for a different initialization of the linking variables and for larger number of blocks—however, the computation times are longer compared to those in Table 4. Thus, we suggest to use 2 blocks as a default value for the heuristic if no instance-specific knowledge is at hand.

Comparing the results of the PADM and Gurobi no clear trend is visible for which kind of instances PADM or Gurobi performs better. For instance, PADM is the faster method for instances like scm11 and scm12 that only contain relatively few integer variables (189 and 210, respectively; cf. Table 3) but is also faster for instance scm40 that contains more than 9600 integers. Instances with many integer variables are usually solved to a better objective value using Gurobi. On the other hand, the table slightly indicates that PADM is the better performing method if an instance contains a large number of nonzero entries and if the solution is rather sparse. However, the latter cannot be checked a priori.

If Gurobi is applied with default settings, i.e., using MIPFocus =0, it is performing slightly worse compared to the results in tables. For example, no solution is found for instance scm35 within the time limit.

Finally, note that the solution time of the PADM is strongly dominated by the running time required by hmetis for computing the decomposition of the instance. The PADM itself is typically very fast. This property is particularly of interest if a proper decomposition of an instance is known by the modeler. In this case, no general-purpose decomposition software need to be applied and the overall approach will be much faster than with using hmetis. However, let us note that the presented results are given for the case of the PADM as a general-purpose heuristic and—even in this case—the results are comparable to the ones of the commercial solver Gurobi.

6. Conclusion

In this paper, we propose a tailored penalty alternating direction method for solving block-decomposed mixed-integer supply chain instances. The sparsity of these instances and the well known fact that they contain many blocks of variables and constraints that are only coupled by a few linking variables are naturally exploited by the method, which (i) breaks down the instance into several blocks, (ii) solves these blocks separately, and that (iii) uses a penalty framework to put the separate block solutions together to a feasible point of the original problem. Our computational results show that the performance of the method used as a general-purpose heuristic, i.e., without a user-specified decomposition, is competitive with commercial MIP solvers. Thus, the method can, in principle, be used standalone as well as a heuristic within an exact branch-and-cut code to speed up the solution process by finding feasible points of good quality quickly. Another option might be to run the heuristic as a standalone heuristic in parallel to Gurobi and use the first solution found. Both

aspects get even more pronounced if a problem-specific decomposition is at hand, e.g., by using structural knowledge of the instances that are often available for the modeler. In this case, the time-consuming general-purpose decomposition obtained by using hmetis can be replaced by directly using a problem-specific decomposition. Finally, the method can also be parallelized in a rather natural way, which might lead to further computational speed-ups for very large supply chain instances.

Acknowledgments

We thank the Deutsche Forschungsgemeinschaft for their support within projects A05, B07, and B08 in the "Sonderforschungsbereich/Transregio 154 Mathematical Modelling, Simulation and Optimization using the Example of Gas Networks" and our industry partner SAP for providing us with real-world SNP instances.

References

[1] M. Bergner, A. Caprara, A. Ceselli, F. Furini, M. E. Lübbecke, E. Malaguti, and E. Traversi. "Automatic Dantzig–Wolfe reformulation of mixed integer programs." In: *Mathematical Programming* 149.1 (2015), pp. 391–424. DOI: 10.1007/s10107-014-0761-5.

[2] L. Bertacco, M. Fischetti, and A. Lodi. "A feasibility pump heuristic for general mixed-integer problems." In: *Discrete Optimization* 4.1 (2007). Mixed Integer Programming IMA Special Workshop on Mixed-Integer Programming, pp. 63–76. DOI: 10.1016/j.disopt.2006.10.001.

[3] R. Borndörfer, C. Ferreira, and A. Martin. "Decomposing Matrices into Blocks." In: *SIAM Journal on Optimization* 9.1 (1998), pp. 236–269. DOI: 10.1137/S1052623497318682.

[4] M. Fischetti, F. Glover, and A. Lodi. "The feasibility pump." In: *Mathematical Programming* 104.1 (2005), pp. 91–104. DOI: 10.1007/s10107-004-0570-3.

[5] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. "Progress in presolving for mixed integer programming." In: *Mathematical Programming Computation* 7.4 (2015), pp. 367–398. DOI: 10.1007/s12532-015-0083-5.

[6] G. Gamrath, A. Gleixner, T. Koch, M. Miltenberger, D. Kniasew, D. Schlögel, A. Martin, and D. Weninger. "Tackling Industrial-Scale Supply Chain Problems by Mixed-Integer Programming." In: *Journal of Computational Mathematics* 37.6 (2019), pp. 866–888. DOI: https://doi.org/10.4208/jcm.1905-m2019-0055. URL: http://global-sci.org/intro/article_detail/jcm/13380.html.

[7] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt. "Penalty Alternating Direction Methods for Mixed-Integer Optimization: A New View on Feasibility Pumps." In: *SIAM Journal on Optimization* 27.3 (2017), pp. 1611–1636. DOI: 10.1137/16M1069687.

[8] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt. "Solving Highly Detailed Gas Transport MINLPs: Block Separability and Penalty Alternating Direction Methods." In: *INFORMS Journal on Computing* 30.2 (2018), pp. 309–323. DOI: 10.1287/ijoc.2017.0780.

[9] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt. "Solving power-constrained gas transportation problems using an MIP-based alternating direction method." In: *Computers & Chemical Engineering* 82 (2015), pp. 303–317. DOI: 10.1016/j.compchemeng.2015.07.005.

[10] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual, Version 8.0.1.* 2018.

[11] A. C. Hax and H. C. Meal. "Hierarchical integration of production planning and scheduling." In: *In Studies in Management Sciences, Vol. 1: Logistics*. Elsevier, 1975. URL: http://hdl.handle.net/1721.1/1868.

[12] G. Karypis and V. Kumar. "Multilevel k-way Hypergraph Partitioning." In: *In Proceedings of the Design and Automation Conference*. 1998, pp. 343–348.

[13] T. Kleinert and M. Schmidt. "Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method." In: *INFORMS Journal on Computing* (2019). URL: http://www.optimization-online.org/DB_HTML/2019/03/7117.html. Forthcoming.

[14] M. Kruber, M. E. Lübbecke, and A. Parmentier. "Learning When to Use a Decomposition." In: *Integration of AI and OR Techniques in Constraint Programming*. Ed. by D. Salvagnin and M. Lombardi. Springer International Publishing, 2017, pp. 202–210.

[15] A. Martin. "Integer programs with block structure." Habilitation thesis. Zuse Institute Berlin, 1999.

[16] R. K. Oliver and M. D. Webber. "Supply-chain management: logistics catches up with strategy." In: *Logistics: The Strategic Issues* (1992).

[17] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[18] H. Stadtler. "Supply chain management and advanced planning–basics, overview and challenges." In: *European Journal of Operational Research* 163.3 (2005), pp. 575–588. URL: http://EconPapers.repec.org/RePEc:eee:ejores:v:163:y:2005:i:3:p:575-588.

[19] B. Yildiz, N. Boland, and M. Savelsbergh. *Decomposition Branching for Mixed Integer Programming*. Tech. rep. 2018. URL: http://www.optimization-online.org/DB_HTML/2018/08/6788.html.

Further Numerical Results

Table 5. Numerical results for the case of 2 blocks, zero objective function coefficients, and using the LP relaxation for initializing the linking variables. All times are given in seconds.

	Gurobi 8.0.1		**PADM**			
ID	obj. value	time	obj. value	$\|\mathcal{L}\|$	h-time	time
scm01	4.03×10^{9}	6	4.03×10^{9}	514	3	6
scm02	1.17×10^{10}	22	$\mathbf{8.28 \times 10^{9}}$	851	12	22
scm03	6.58×10^{9}	9	6.58×10^{9}	771	5	9
scm04	8.60×10^{9}	13	8.60×10^{9}	1028	7	13
scm05	1.64×10^{10}	30	1.64×10^{10}	2026	16	29
scm06	1.38×10^{10}	21	1.38×10^{10}	1541	11	21
scm07	2.59×10^{10}	41	2.59×10^{10}	2568	21	41
scm08	5.04×10^{10}	110	5.04×10^{10}	4998	55	110
scm09	$\mathbf{4.44 \times 10^{11}}$	95	4.61×10^{11}	601	201	273
scm10	$\mathbf{5.76 \times 10^{12}}$	343	1.23×10^{13}	877	502	717
scm11	$\mathbf{1.23 \times 10^{9}}$	4	2.74×10^{9}	84	2	4
scm12	3.44×10^{11}	6	$\mathbf{9.22 \times 10^{9}}$	105	3	6
scm13	$\mathbf{7.14 \times 10^{8}}$	5	1.34×10^{9}	105	2	5
scm14	2.05×10^{12}	34	$\mathbf{2.42 \times 10^{9}}$	336	15	34
scm15	–	67	$\mathbf{8.60 \times 10^{9}}$	578	29	67
scm16	$\mathbf{4.85 \times 10^{9}}$	33	9.51×10^{9}	42	15	33
scm17	1.59×10^{16}	2	$\mathbf{6.43 \times 10^{15}}$	835	1	2
scm18	–	50	$\mathbf{1.70 \times 10^{17}}$	2640	14	50
scm19	$\mathbf{3.19 \times 10^{12}}$	68	1.95×10^{13}	2360	29	68
scm20	–	10	$\mathbf{5.39 \times 10^{14}}$	688	5	10
scm21	$\mathbf{8.84 \times 10^{8}}$	11	9.00×10^{9}	1751	3	11
scm22	$\mathbf{1.74 \times 10^{9}}$	10	1.15×10^{10}	2184	5	10
scm23	$\mathbf{2.67 \times 10^{9}}$	11	1.71×10^{10}	2850	8	11
scm24	$\mathbf{-2.91 \times 10^{8}}$	3	-2.45×10^{8}	1004	1	3
scm25	$\mathbf{-3.28 \times 10^{8}}$	4	-2.74×10^{8}	762	2	4
scm26	$\mathbf{-5.75 \times 10^{8}}$	7	-4.83×10^{8}	734	3	7
scm27	$\mathbf{5.21 \times 10^{14}}$	7	6.90×10^{17}	1159	4	7
scm28	$\mathbf{5.21 \times 10^{14}}$	10	2.08×10^{18}	2072	7	10
scm29	$\mathbf{6.80 \times 10^{14}}$	18	2.00×10^{18}	3472	12	18
scm30	$\mathbf{2.45 \times 10^{11}}$	14	3.28×10^{11}	55	17	26
scm31	–	59	$\mathbf{6.81 \times 10^{11}}$	363	30	59
scm32	$\mathbf{5.89 \times 10^{11}}$	5	1.34×10^{12}	115	3	5
scm33	–	23	$\mathbf{9.60 \times 10^{12}}$	629	13	23
scm34	–	56	$\mathbf{1.91 \times 10^{13}}$	1644	33	56
scm35	$\mathbf{6.25 \times 10^{11}}$	24	2.74×10^{12}	332	17	24
scm36	$\mathbf{2.70 \times 10^{9}}$	14	1.80×10^{10}	2853	8	14
scm37	$\mathbf{-5.76 \times 10^{8}}$	6	-5.12×10^{8}	464	3	6
scm38	$\mathbf{5.98 \times 10^{15}}$	24	1.26×10^{18}	3478	12	24
scm39	$\mathbf{2.47 \times 10^{11}}$	62	6.92×10^{11}	441	30	62
scm40	–	58	$\mathbf{1.37 \times 10^{13}}$	1632	39	58

(L. Schewe) The University of Edinburgh, School of Mathematics, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, UK
Email address: lars.schewe@ed.ac.uk

(D. Weninger) Friedrich-Alexander-Universität Erlangen-Nürnberg, Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany
Email address: dieter.weninger@fau.de

(M. Schmidt) Trier University, Department of Mathematics, Universitätsring 15, 54296 Trier, Germany
Email address: martin.schmidt@uni-trier.de

TABLE 6. Numerical results for the case of 4 blocks, zero objective function coefficients, and 0 as initial values for the linking variables. All times are given in seconds.

	Gurobi 8.0.1		**PADM**			
ID	obj. value	time	obj. value	$\|\mathcal{L}\|$	h-time	time
scm01	4.03×10^{9}	6	4.03×10^{9}	815	5	6
scm02	1.17×10^{10}	21	$\mathbf{8.28 \times 10^{9}}$	2371	18	21
scm03	6.58×10^{9}	10	6.58×10^{9}	1183	9	10
scm04	8.60×10^{9}	13	8.60×10^{9}	1595	11	13
scm05	1.64×10^{10}	30	1.64×10^{10}	3077	26	30
scm06	1.38×10^{10}	22	1.38×10^{10}	2366	18	22
scm07	2.59×10^{10}	39	2.59×10^{10}	3928	34	39
scm08	5.04×10^{10}	96	5.04×10^{10}	7702	86	96
scm09	$\mathbf{4.44 \times 10^{11}}$	104	4.61×10^{11}	2113	324	347
scm10	$\mathbf{5.76 \times 10^{12}}$	320	1.60×10^{13}	1818	827	873
scm11	$\mathbf{1.23 \times 10^{9}}$	5	2.24×10^{9}	397	4	5
scm12	3.44×10^{11}	6	$\mathbf{9.22 \times 10^{9}}$	732	5	6
scm13	$\mathbf{3.58 \times 10^{8}}$	6	1.34×10^{9}	642	5	6
scm14	2.05×10^{12}	25	$\mathbf{2.42 \times 10^{9}}$	609	23	25
scm15	–	50	$\mathbf{8.58 \times 10^{9}}$	993	45	50
scm16	$\mathbf{4.85 \times 10^{9}}$	30	9.30×10^{9}	676	24	30
scm17	1.37×10^{16}	4	$\mathbf{6.53 \times 10^{15}}$	2237	3	4
scm18	3.27×10^{17}	295	$\mathbf{1.59 \times 10^{17}}$	4561	21	295
scm19	$\mathbf{3.19 \times 10^{12}}$	114	2.17×10^{13}	7190	54	114
scm20	–	9	$\mathbf{3.64 \times 10^{14}}$	1431	7	9
scm21	$\mathbf{8.84 \times 10^{8}}$	19	9.61×10^{9}	3532	6	19
scm22	$\mathbf{1.74 \times 10^{9}}$	16	1.23×10^{10}	4458	9	16
scm23	$\mathbf{2.67 \times 10^{9}}$	19	1.79×10^{10}	5996	14	19
scm24	$\mathbf{-2.91 \times 10^{8}}$	7	-2.45×10^{8}	2826	3	7
scm25	$\mathbf{-3.30 \times 10^{8}}$	10	-2.74×10^{8}	2426	4	10
scm26	$\mathbf{-5.77 \times 10^{8}}$	21	-4.83×10^{8}	2341	7	21
scm27	$\mathbf{5.21 \times 10^{14}}$	8	5.57×10^{16}	2188	6	8
scm28	$\mathbf{5.21 \times 10^{14}}$	14	5.83×10^{16}	4097	11	14
scm29	$\mathbf{6.80 \times 10^{14}}$	38	7.13×10^{17}	6852	19	38
scm30	$\mathbf{2.45 \times 10^{11}}$	14	3.23×10^{11}	98	27	31
scm31	–	52	$\mathbf{6.84 \times 10^{11}}$	647	47	52
scm32	$\mathbf{5.89 \times 10^{11}}$	6	1.47×10^{12}	223	4	6
scm33	–	26	$\mathbf{1.25 \times 10^{13}}$	1019	21	26
scm34	–	60	$\mathbf{2.54 \times 10^{13}}$	2787	48	60
scm35	$\mathbf{6.17 \times 10^{11}}$	33	2.74×10^{12}	1067	28	33
scm36	$\mathbf{2.70 \times 10^{9}}$	21	1.87×10^{10}	6037	13	21
scm37	$\mathbf{-5.76 \times 10^{8}}$	12	-5.12×10^{8}	1392	6	12
scm38	$\mathbf{5.98 \times 10^{15}}$	23	9.75×10^{17}	6431	19	23
scm39	$\mathbf{2.47 \times 10^{11}}$	54	6.87×10^{11}	700	49	54
scm40	–	68	$\mathbf{2.60 \times 10^{13}}$	2695	60	68