Data Supplement for the paper "Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method"

Thomas Kleinert[1,2], Martin Schmidt[3,2]

This note describes the MATLAB function ssdMatrixGenerator, which is used to set up the QP data for some of the numerical experiments in the paper "Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method" by Thomas Kleinert and Martin Schmidt. This method generates symmetric and positive semidefinite matrices H_u and G_u as well as symmetric and negative definite matrices G_l. These matrices are used to extend linear (mixed-integer) bilevel problems from the literature to bilevel problems of the following form:

$$\begin{aligned}
\min_{x,y} \quad & \frac{1}{2}x^\top H_u x + c^\top x + \frac{1}{2}y^\top G_u y + d^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \\
& y \in \arg\max_{\bar{y}} \left\{ \frac{1}{2}\bar{y}^\top G_l \bar{y} + e^\top \bar{y} \colon Cx + D\bar{y} \leq b \right\}.
\end{aligned}$$

The function

density = ssdMatrixGenerator(dim, maxAbsCoeff, definite, sign, filename)

creates a randomly generated symmetric (semi-)definite matrix of size $\mathsf{dim} \times \mathsf{dim}$ with integer entries in $\{-\mathsf{maxAbsCoeff}, \dots, \mathsf{maxAbsCoeff}\}$ and writes it to the file filename. The parameter $\mathsf{definite} \in \{0, 1\}$ specifies whether the resulting matrix is semidefinite ($\mathsf{definite} = 0$) or strictly definite ($\mathsf{definite} = 1$). Further, the parameter $\mathsf{sign} \in \{-1, 1\}$ decides whether the resulting matrix is positive ($\mathsf{sign} = 1$) or negative ($\mathsf{sign} = -1$) (semi-)definite. The generated matrix has the density density.

In the following, we briefly specify the implementation of the function.

1. We generate a random sparse matrix S of size $\mathsf{dim} \times \mathsf{dim}$ and density $\delta(\mathsf{dim})$ in the following way:
   (i) The density $\delta(\mathsf{dim})$ is computed by the piecewise linear function
   $$\delta(\mathsf{dim}) = \delta_{i(\mathsf{dim})} + \frac{\delta_{i(\mathsf{dim})} - \delta_{i(\mathsf{dim})+1}}{d_{i(\mathsf{dim})} - d_{i(\mathsf{dim})+1}} \left(\mathsf{dim} - d_{i(\mathsf{dim})}\right),$$
   where $i(\mathsf{dim}) = \max\{i : d_i \leq \mathsf{dim}\}$. The values d_i and δ_i are hard-coded as specified in Table 1.
   (ii) MATLAB's built-in function randi (that generates uniformly distributed pseudo-random integers) is used to generate three random integer vectors rows, cols, and values of size $\delta(\mathsf{dim}) \cdot \mathsf{dim}^2$. The vectors rows and cols have entries from $\{1, \dots, \mathsf{dim}\}$ and the vector values has entries from $\{-\mathsf{maxAbsCoeff}, \dots, \mathsf{maxAbsCoeff}\}$.
   (iii) Every entry in values equal to 0 is replaced by a random integer in $\{1, \dots, \mathsf{maxAbsCoeff}\}$ using randi.
   (iv) A randomly chosen (using randi) entry in rows is set to dim. The same is done for a randomly chosen entry in cols. This ensures that the resulting random matrix S has indeed size $\mathsf{dim} \times \mathsf{dim}$.

Date: November 5, 2019.

(v) MATLAB's built-in function sparse is used to generate a sparse matrix S from the triplets specified by the vectors rows, cols, and values.

2. Generate a positive (semi-)definite matrix M from the matrix S:
   (i) Compute the symmetric positive semidefinite matrix $\mathsf{M} = \mathsf{S}' \cdot \mathsf{S}$.
   (ii) If definite equals 1, randi is used to generate a random integer vector d of size dim and entries in $\{1, \dots, \mathsf{maxAbsCoeff}\}$. This vector is used to generate a sparse diagonal matrix D using the built-in function spdiags. The matrix $\mathsf{M} \leftarrow \mathsf{M} + \mathsf{D}$ is positive definite.
   (iii) Set $\mathsf{M} \leftarrow \mathsf{sign} \cdot \mathsf{M}$ to obtain a positive ($\mathsf{sign} = 1$) or negative ($\mathsf{sign} = -1$) (semi-)definite matrix.
3. Write M to filename. In filename every line consists of a space-separated triplet "column-index row-index value".

Table 1. Specification of the values d_i and δ_i.

i	1	2	3	4	5	6	7	8
d_i	1	25	50	75	100	250	500	750
δ_i	1.0	1.0	0.5	0.2	0.15	0.06	0.036	0.027
i	9	10	11	12	13	14	15	16
d_i	1000	2500	5000	7500	10000	15000	20000	30000
δ_i	0.022	0.0135	0.009	0.0072	0.006	0.0045	0.0035	0.0025
i	17	18	19	20	21	22	23	24
d_i	40000	50000	60000	70000	80000	90000	100000	125000
δ_i	0.0015	0.0011	0.0008	0.0006	0.0005	0.0004	0.00025	0.00022
i	25	26	27	28	29	30	31	32
d_i	150000	175000	200000	250000	500000	1000000	1500000	2000000
δ_i	0.00018	0.00014	0.0001	0.00005	0.00001	0.000005	0.000001	0.000001

(T. Kleinert, M. Schmidt) [1]Friedrich-Alexander-Universität Erlangen-Nürnberg, Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany; [2]Energie Campus Nürnberg, Fürther Str. 250, 90429 Nürnberg, Germany; [3]Trier University, Department of Mathematics, Universitätsring 15, 54296 Trier, Germany

Email address: thomas.kleinert@fau.de

Email address: martin.schmidt@uni-trier.de