ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.2.019

MINIMUM-VOLUME COVERING ELLIPSOIDS: IMPROVING THE EFFICIENCY OF THE WOLFE-ATWOOD ALGORITHM FOR LARGE-SCALE INSTANCES BY POOLING AND BATCHING

Jakub Kůdela

Institute of Automation and Computer Science, Brno University of Technology, Czech Republic
Jakub.Kudela@vutbr.cz

Abstract

The Minimum-Volume Covering Ellipsoid (MVCE) problem is an important optimization problem that comes up in various areas of engineering and statistics. In this paper, we improve the state-of-the-art Wolfe-Atwood algorithm for solving the MVCE problem with pooling and batching procedures. This implementation yields significant improvements on the runtime of the algorithm for large-scale instances of the MVCE problem, which is demonstrated on quite extensive computational experiments.

Keywords: *minimum-volume covering ellipsoid, Löwner-John ellipsoid, large-scale optimization, Wolfe-Atwood algorithm, pooling, batching.*

Received: 02 October 2019
Accepted: 10 December 2019
Published: 21 December 2019

1 Introduction

Let $\mathcal{X} = \{x_1, \ldots, x_m\} \subset \mathcal{R}^n$ be a finite set of vectors whose affine hull is $\mathcal{R}^n$. In this paper, the problem in question will be finding the Minimum-Volume Covering Ellipsoid (MVCE) of $\mathcal{X}$, also known as the Löwner-John ellipsoid. The recently published book [1] gives a wonderful overview of the MVCE problem, and will serve as our primary source for the introduction and the apparatus for handling the MVCE problem. The idea of MVCE and the proofs of its existence and uniqueness come from Löwner (unfortunately in an unpublished work). The first published results about MVCE appear in the 1950's in [2] and [3]. An earlier result by John [4] about the suitability of ellipsoids for fitting convex compact sets was also one of the pivotal starting points for the study of the MVCE problem (hence, the alternative name for the problem is deciphered!). The MVCE problem comes up in several applied and theoretical areas. It has a strong connection to the optimal design in statistics [5]. Other application in statistics include outlier identification (see [6], [7], and [8]), and clustering (see [9], [10], and [11]). Containing ellipsoids play an important role in robust optimization [12] and control theory [13]. MVCEs are also used in computational geometry and computer graphics [14], e.g. for obstacle avoidance in robotics. One of the standard methods for solving the MVCE problem, the Frank–Wolfe method, is being investigated for its suitability for solving statistical learning problems [15, 16].

The main contribution of this paper is in the efficient computation of large scale instances of the MVCE problem (in both the dimension n and the number of vectors m). Our starting point is the Wolfe-Atwood (WA) method [17, 18] with Kumar-Yıldırım initialization [19]. We describe a pooling scheme, in which we disregard a large portion of the vectors, solving the MCVE problem for only a small subset of $\mathcal{X}$. Once a solution of this surrogate problem is found, we look at the whole $\mathcal{X}$ and find the points that were the furthest away from the ellipsoid. These points are then added to the surrogate problem. This procedure is repeated until a "good enough" solution is found (the precise conditions will be defined in the following sections). A similar idea in the context of chance constrained optimization problems was investigated in [20] and [21].

2 The MVCE Problem

There are several equivalent mathematical definitions of ellipsoids. The one adopted in this paper is the following: An ellipsoid $\mathcal{E}(H, \bar{x})$ is a set of the form

$$\mathcal{E}(H, \bar{x}) = \{x \in \mathcal{R}^n : (x - \bar{x})^T H (x - \bar{x}) \leq n\},$$

where $\bar{x} \in \mathcal{R}^n$ is the center of the ellipsoid and H is a symmetric positive definite matrix of order n. Another way of thinking about ellipsoids is through affine transformations of balls in $\mathcal{R}^n$. In particular, let L be the Cholesky factor of H, i.e. $H = LL^T$, with L being a lower triangular matrix with positive diagonal entries. Then x lies in $\mathcal{E}(H, \bar{x})$ iff $||L^T(x - \bar{x})|| \leq \sqrt{n}$, so that

$$\mathcal{E}(H, \bar{x}) = \{x = \bar{x} + (\sqrt{n} L^{-T}) z : z \in \mathcal{R}^n, ||z|| \leq 1\}.$$

The volume of the ellipsoid can therefore be computed as the volume of the unit ball times $|\det(\sqrt{n}L^{-T})|$. Using the fact that $H = LL^T$, and $\det H = (\det L)^2$, we get

$$\mathrm{vol}(\mathcal{E}(H, \bar{x})) = \frac{n^{n/2}\Omega_n}{\sqrt{\det H}},$$

where Ω_n is the volume of a unit ball in $\mathcal{R}^n$. This means that if we seek to minimize the volume of the ellipsoid, we can frame it as a minimization of the negative of the determinant of its shape matrix H. The problem of finding the MVCE of $\mathcal{X}$ then attains the following form:

$$\begin{array}{ll} \underset{H\in\mathcal{S}_+^n,\bar{x}\in\mathcal{R}^n}{\text{minimize}} & -\ln\det(H) \\ \text{subject to} & (x_i-\bar{x})^T H(x_i-\bar{x}) \le n, \quad i=1,\dots,m, \end{array} \tag{1}$$

where $\mathcal{S}_+^n$ is the space of symmetric positive definite matrices of order n. In the problem (1) the "centering" variable $\bar{x}$ causes computational difficulties [1]. These difficulties can be circumvented by "lifting" the MVCE problem to a higher dimension and solving an equivalent problem, that has $\bar{x} = 0$, see [22]. This equivalent problem is then convex [23] and has the following form:

$$\begin{array}{ll} \underset{H\in\mathcal{S}_+^n}{\text{minimize}} & -\ln\det(H) \\ \text{subject to} & x_i^T H x_i \le n, \quad i=1,\dots,m. \end{array} \tag{2}$$

We say that H is feasible for (2) if it satisfies all m constraints and is positive definite. Denoting $u_i \ge 0$ the Lagrange multiplier for the ith constraint, the Lagrangian $L(H,u)$ for (2) is

$$L(H,u) = -\ln\det(H) + \sum_{i=1}^{m} u_i(x_i^T H x_i).$$

If we denote by e the vector of ones in $\mathcal{R}^m$, by U the diagonal matrix from the values of u:

$$U = \mathrm{Diag}(u) \in S^m,$$

and by X the matrix $X = [x_1,\dots,x_m] \in \mathcal{R}^{n\times m}$, we can rewrite the Lagrangian as

$$L(H,u) = -\ln\det(H) + HXUX^T - ne^T u.$$

After a series of manipulations (see [1]) we obtain a dual of (2):

$$\begin{array}{ll} \underset{u\in\mathcal{R}^m}{\text{maximize}} & \ln\det(XUX^T) \\ \text{subject to} & e^T u = 1, \\ & u \ge 0. \end{array} \tag{3}$$

If the affine hull of $\mathcal{X}$ is $\mathcal{R}^n$ (or, equivalently, if X has full row rank), the optimal solutions to (2) and (3) are unique and attain the same optimal objective value [1] (i.e., strong duality holds).

Another important result that concerns our investigation is the following: there is a finite subset of $\mathcal{X}$ of cardinality at most $n(n+1)/2$, such that the MVCE containing this subset is also the MVCE for the whole set $\mathcal{X}$ [1]. This small subset is called a core set. The implication being that the optimal solution to (3) has only $n(n+1)/2$ positive components. Notice that this number does not depend on the number of points m.

Since these problems cannot usually be solved exactly, we will be interested in finding ϵ-primal feasible or ϵ-approximately optimal solutions: A feasible u for the problem (3) is said to be ϵ-primal feasible if $H(u) = (XUX^T)^{-1}$ satisfies

$$x_i^T H(u) x_i \le (1+\epsilon)n, \quad i=1,\dots,m.$$

Additionally, if it satisfies

$$x_i^T H(u) x_i \ge (1-\epsilon)n \text{ if } u_i > 0, \quad i=1,\dots,m,$$

we say that u is ϵ-approximately optimal.

Algorithm 1 WA Algorithm

procedure STEP 0:
  Choose u feasible for (3) and $\epsilon > 0$.
  Compute $\omega = \omega(u)$ and a (scaled) Cholesky factorization of XUX^T.
  Go to Step 1.
procedure STEP 1:
  Given the current iterate u and its associated $\omega = \omega(u)$:
  Compute $\epsilon_+ = \max_h\{(\omega_h - n)/n\}$, with $h = i$ attaining the maximum.
  Compute $\epsilon_- = \max_h\{(n - \omega_h)/n : u_h > 0\}$, with $h = j$ attaining the maximum.
  if $\max \epsilon_+, \epsilon_- \leq \epsilon$ **then** STOP: u is ϵ-approximately optimal.
  else
    if $\epsilon_+ > \epsilon_-$ **then** go to Step 2;
    else go to Step 3.
procedure STEP 2:
  Compute $\lambda^* = \frac{\omega_i - n}{(n+1)\omega_i}$ and update $u \leftarrow (1 + \lambda^*)^{-1}(u + \lambda^* e_i)$.
  Go to Step 4.
procedure STEP 3:
  Compute $\lambda^* = \frac{\omega_j - n}{(n+1)\omega_j}$ and set $\lambda = \max\{-u_j, \lambda^*\}$.
  Update $u \leftarrow (1 + \lambda)^{-1}(u + \lambda e_j)$.
  Go to Step 4.
procedure STEP 4:
  Update ω and a scaled Cholesky factorization of XUX^T.
  Go to Step 1.

3 Solving the MVCE Problem

The primary approaches for solving the MVCE problem have been first-order methods (a modified Frank-Wolfe method [24]) applied to the dual problem (3), denoted here as the WA method [17, 18]. There exists a similar method for solving the MVCE problem attributed to Fedorov [25] and Wynn [26] that was proposed at around the same time as the WA method. However, this method was found computationally inferior to the WA method in several numerical studies [27]. A parallel line of research was conducted in second-order methods for solving the primal problem (2). The resulting dual reduced Newton method with an active-set strategy [28] was successful in being faster than the WA method [27], but could not be used for truly large-scale problems because of its memory requirements (for problems in higher dimensions n).

In our baseline implementation, we will use the WA method, with its form is taken from [1]. The individual steps are described in Algorithm 1. An important factor in the efficiency of the WA algorithm is the starting value of u. Notable initialization schemes are the due to Khachiyan [29], and Kumar and Yıldırım [19]. The Kachiyan initialization is quite straightforward, choosing $u = e/m$. The Kumar and Yıldırım (KY) strategy is a little bit more involved (and also work a bit better), and is described in Algorithm 2. The Steps 0 through 2 of the algorithm are due to [30]. Another increase in the effectiveness of the implementation of the WA method can be made by eliminating points that do not belong to the core set. The method for carrying out the elimination was developed in [31] and the positive impact on the runtime of the algorithm was investigated in [32].

3.1 WA with Pooling and Batching

In a similar fashion to the point elimination [31], our idea of improving the efficiency of WA centers around the core set. However, instead of reducing the size of the solved problem by eliminating points that do not belong to the core set, we take what might be conceived as the direct opposite approach. We devise a surrogate problem that is considerably smaller – a subset $\tilde{\mathcal{X}}$ of $\mathcal{X}$ – find the MVCE of $\tilde{\mathcal{X}}$ by Algorithm 1, and check, whether the MVCE of $\tilde{\mathcal{X}}$ contains all the points of $\mathcal{X}$. If it does, we are done – $\tilde{\mathcal{X}}$ had to contain the core set of $\mathcal{X}$ and the MVCE of the two sets is identical. If, however, some points of $\mathcal{X}$ lie outside the MCVE of $\tilde{\mathcal{X}}$, the core sets must differ and we need to update $\tilde{\mathcal{X}}$. In the update, we find the points in $\mathcal{X}$ that are the furthers away from the MVCE of $\tilde{\mathcal{X}}$. The maximum number of points that are added during the update is called a batch size and is denoted by k. Effectiveness of batching was previously investigated, in the context of stochastic programming, in [33]. Moreover, we found it computationally advantageous to use the resulting u's from the computations of MVCE of $\tilde{\mathcal{X}}$ as starting points for the updated iterations. The individual steps are summarized in Algorithm 3.

The selection of points that will be added to the surrogate problem $\tilde{\mathcal{X}}$ requires the computation of the

Algorithm 2 KY Initialization

procedure STEP 0:
  Choose an arbitrary nonzero $c_1 \in \mathcal{R}^n$ and set $j = 1$.
  Go to Step 1.
procedure STEP 1:
  Let $\bar{z}_j$ and $\underline{z}_j$ maximize and minimize $c_j^T x$ over the x_h's. Set $y_j = \bar{z}_j - \underline{z}_j$.
  Go to Step 2.
procedure STEP 2:
  if $j < n$ **then** choose an arbitrary nonzero c_{j+1} orthogonal to $y_1, \ldots, y_j$
    increase j by 1, and go to Step 1.
  else go to Step 3.
procedure STEP 3:
  Compute u by putting equal weight on each of the distinct points $Z = \{\bar{z}_j, \underline{z}_j : j = 1, \ldots, n\}$.
  STOP.

Algorithm 3 WA with Pooling and Batching

procedure STEP 0:
  Choose a batch size k, a subset $\tilde{\mathcal{X}}$ of $\mathcal{X}$, and $\epsilon > 0$.
  Set the starting values $\tilde{u}_S$ for computing the MVCE of $\tilde{\mathcal{X}}$ according to Algorithm 2.
  Go to Step 1.
procedure STEP 1:
  Find the (ϵ-approximately optimal) solution of the MVCE problem of $\tilde{\mathcal{X}}$
  using Algorithm 1, with starting values $\tilde{u}_S$.
  Get the Cholesky factorization LL^T of $\tilde{X}\tilde{U}\tilde{X}^T$ and the optimal $\tilde{u}$ (these are produced by Algorithm 1).
  Go to Step 2.
procedure STEP 2:
  for $x_i \in \mathcal{X}$ **do**
    $v(x_i) = ||L^{-T} x_i||^2 - n$
  Compute the set of points in $\mathcal{X}$ not covered by the MVCE of $\tilde{\mathcal{X}}$: $\mathcal{X}_v = \{x_i : v(x_i) > 0\}$.
  if $\mathcal{X}_v = \emptyset$ **or** $\mathcal{X}_v \subseteq \tilde{\mathcal{X}}$ **then** STOP:
  the core sets are the same and the ϵ-approximately optimal solution u to
  the whole MVCE problem can be constructed from $\tilde{u}$.
  else find the k points of $\mathcal{X}_v$ with the highest values of $v(x_i)$: $\tilde{\mathcal{X}}_v = \text{maxk}_{x_i} v(x_i)$.
    Set $\tilde{u}_S = \tilde{u}$. Update $\tilde{\mathcal{X}} \leftarrow \tilde{\mathcal{X}} \cup \tilde{\mathcal{X}}_v$. Append zeros to $\tilde{u}_S$ to match the size of $\tilde{\mathcal{X}}$.
    Go to Step 1.

following quantity

$$||L^{-T} x_i||^2 - n$$

for each $x_i \in \mathcal{X}$. Since L comes from a Cholesky factorization it is an upper-triangular matrix, and the computation of the inverse does not pose significant computational burden. When implementing Step 2 of Algorithm 3, one should be careful about properly handling the optimal multipliers $\tilde{u}$. We suppose that the newly added points to $\tilde{\mathcal{X}}$ will be appended to the corresponding matrix $\tilde{X}$, hence the instruction to append the corresponding number of zeros to $\tilde{u}_S$. If, for whatever reason, one decided to first somehow sort the individual vectors $x_i \in \tilde{\mathcal{X}}$ before constructing the matrix $\tilde{X}$, the added zeros to the starting values $\tilde{u}_S$ would appear on different places (not simply appended).

4 Computational Experiments

Compared to the WA (Algorithm 1), our Algorithm 3 uses two additional "hyperparameters" – the size of the batch k, and the size (an selection strategy) of the starting subset $\tilde{X}$. In the following computational experiments, we will vary only the batch size parameter k, leaving the investigation into the impact of the selection strategy of $\tilde{\mathcal{X}}$ as our future work. We will use two different setting for the computational experiments, both involving random generation of the set $\mathcal{X}$ and centered around the origin.

In the first setting each of the m vectors $x \in \mathcal{X} \subset \mathcal{R}^n$ follows a multivariate normal distribution $x \sim \mathcal{N}(0, \Sigma)$, where the individual values $\sigma_{i,j}$ of the covariance matrix Σ follow a standard normal distribution $\sigma_{i,j} \sim \mathcal{N}(0, 1)$, $i, j = 1, \ldots, n$. This configuration, however, results in the majority of points lying close to the surface of an ellipsoid (whose shape is guided by Σ), which is quite a special situation [1].

Table 1: The results from the computational experiments, the first setting. Geometric average of runtimes in seconds over 10 independent runs. The experiments with "-" were deemed not computationally interesting.

			batch size										
n	m	WA	10	20	50	100	200	500	1,000	2,000	5,000	10,000	speedup
5	1,000	0.005	0.015	0.017	0.014	0.016	0.018	0.014	-	-	-	-	0.382
	5,000	0.012	0.025	0.023	0.024	0.021	0.020	0.026	0.028	0.028	-	-	0.594
	10,000	0.017	0.034	0.034	0.029	0.025	0.028	0.030	0.028	0.030	0.040	-	0.681
	50,000	0.046	0.055	0.053	0.044	0.039	0.045	0.033	0.043	0.046	0.055	0.064	1.372
	100,000	0.052	0.062	0.058	0.047	0.045	0.042	0.043	0.042	0.047	0.054	0.068	1.234
	500,000	0.189	0.179	0.144	0.147	0.134	0.119	0.110	0.085	0.096	0.131	0.145	2.242
	1,000,000	0.360	0.289	0.260	0.235	0.203	0.198	0.166	0.149	0.150	0.168	0.191	2.419
	5,000,000	1.585	1.194	1.025	0.851	0.806	0.749	0.658	0.622	0.549	0.535	0.578	2.965
10	1,000	0.019	0.059	0.055	0.046	0.043	0.042	0.049	-	-	-	-	0.440
	5,000	0.046	0.126	0.120	0.107	0.092	0.086	0.093	0.118	0.115	-	-	0.541
	10,000	0.043	0.096	0.104	0.099	0.077	0.101	0.072	0.079	0.083	0.103	-	0.588
	50,000	0.116	0.215	0.201	0.168	0.137	0.161	0.133	0.108	0.132	0.158	0.194	1.080
	100,000	0.185	0.216	0.196	0.174	0.181	0.138	0.132	0.107	0.119	0.163	0.185	1.731
	500,000	1.015	0.585	0.478	0.378	0.332	0.299	0.290	0.281	0.225	0.251	0.323	4.503
	1,000,000	1.968	0.923	0.761	0.578	0.525	0.466	0.451	0.385	0.331	0.328	0.371	6.004
	5,000,000	9.735	4.031	2.893	2.409	2.054	1.758	1.684	1.595	1.334	1.154	1.218	8.432
20	1,000	0.042	0.193	0.139	0.140	0.120	0.090	0.117	-	-	-	-	0.461
	5,000	0.085	0.338	0.267	0.222	0.209	0.197	0.178	0.201	0.204	-	-	0.476
	10,000	0.135	0.439	0.356	0.300	0.276	0.256	0.269	0.210	0.262	0.324	-	0.642
	50,000	0.402	0.751	0.521	0.464	0.413	0.405	0.410	0.362	0.296	0.418	0.541	1.359
	100,000	0.704	0.887	0.635	0.503	0.473	0.483	0.485	0.459	0.391	0.431	0.589	1.801
	500,000	3.484	2.148	1.479	1.073	0.933	0.822	0.761	0.757	0.733	0.677	0.846	5.146
	1,000,000	6.441	3.524	2.342	1.632	1.374	1.259	1.031	1.068	1.109	1.018	0.970	6.642
	5,000,000	30.038	14.701	9.619	6.568	5.238	4.579	3.865	3.426	3.559	3.288	2.735	10.983
50	1,000	0.147	1.479	0.879	0.502	0.415	0.443	0.386	-	-	-	-	0.381
	5,000	0.437	3.260	1.909	1.097	0.891	0.837	0.856	0.722	0.879	-	-	0.605
	10,000	0.729	4.273	2.504	1.409	1.108	1.013	1.032	1.089	1.016	1.514	-	0.719
	50,000	4.667	6.930	3.882	2.150	1.782	1.612	1.667	1.804	1.895	2.082	2.579	2.894
	100,000	8.847	9.691	5.351	3.035	2.231	2.156	2.208	2.153	2.417	2.412	3.088	4.108
	500,000	44.959	21.683	12.035	6.197	4.289	3.616	3.525	3.362	3.868	4.568	5.488	13.371
	1,000,000	89.149	35.820	19.709	10.040	7.015	5.829	4.849	4.764	4.962	6.040	7.074	18.714
	5,000,000	431.116	144.452	77.452	38.096	25.061	18.624	15.171	13.684	12.471	14.772	15.323	34.569
100	1,000	0.398	8.760	4.690	2.285	1.461	1.201	0.826	-	-	-	-	0.482
	5,000	1.781	29.775	16.046	7.467	4.747	3.408	3.436	3.716	3.484	-	-	0.523
	10,000	5.048	42.860	22.835	10.670	6.584	4.735	4.514	4.780	5.031	6.466	-	1.118
	50,000	30.095	84.405	45.317	20.866	12.846	8.938	7.905	9.080	10.165	14.547	21.159	3.807
	100,000	57.330	106.067	56.866	25.921	15.561	10.986	9.318	11.071	12.246	18.836	27.353	6.152
	500,000	307.890	216.461	113.914	50.910	29.678	19.930	16.115	17.438	18.294	29.148	48.038	19.106
	1,000,000	609.408	315.621	164.033	72.753	41.914	28.012	20.927	22.564	24.355	36.033	56.474	29.120
	5,000,000	3,033.0	1,043.3	540.582	232.550	129.229	80.630	56.175	51.499	52.057	61.538	84.750	58.896
200	1,000	0.853	30.127	15.635	6.986	3.840	2.205	1.538	-	-	-	-	0.555
	5,000	10.174	286.527	150.195	65.552	36.488	22.073	14.288	13.784	16.305	-	-	0.738
	10,000	27.303	496.009	261.568	114.592	63.067	38.230	26.362	29.361	48.044	58.744	-	1.036
	50,000	139.740	1,304.1	688.714	302.637	168.061	101.514	68.658	71.244	112.661	153.901	154.094	2.035
	100,000	284.813	1,932.2	1,002.5	449.379	252.378	151.219	104.397	110.783	141.236	194.298	232.266	2.728
	500,000	1,491.5	-	-	947.973	527.926	317.777	201.206	191.190	227.911	264.674	369.161	7.801
	1,000,000	3,000.4	-	-	1,329.8	731.252	426.846	254.428	228.103	279.087	310.343	434.680	13.154
	5,000,000	15,678	-	-	3,204.3	1,720.9	978.962	539.251	422.558	414.344	523.700	632.775	37.839

In the second setting, we will generate instances that should not have the majority of points close to an ellipsoidal boundary (this procedure is, yet again, taken from [1]). We generate m independent standard Cauchy random variable and an $n \times m$ matrix A of independent standard normal variables. Then we set each point x_i be the normalized ith column of A times the ith Cauchy sample. The points generated by this procedure satisfy rotational symmetry, and their distances from the origin are Cauchy.

In the computational experiments, we compare the WA method with the KY initialization and point elimination (the implementation is taken directly from [1]) to our method described in Algorithm 3. We set the optimality parameter $\epsilon = 10^{-7}$ for both methods. The selection of $\tilde{\mathcal{X}}$ is quite straightforward – we simply pick the first n vectors of $\mathcal{X}$. We implemented the algorithms in MATLAB 2019a and used a machine with a six-core AMD Ryzen 5 2600X 3.6 GHz processor and 32 GB RAM.

The magnitude of computational experiments was quite extensive. We varied the number of vectors m between 1,000 and 5,000,000, the number of dimension n between 5 and 200, and the size of the batch k between 10 and 10,000. Each of the experiments (for different values of m, n, and k) was carried out 10 times (in both settings). The results are summarized in Table 1 (for the first setting) and Table 2 (for the second one).

It is apparent from the results, that the pooling and batching procedures described in Algorithm 3 can substantially decrease the computational efforts for solving the MVCE problem for the truly large-scale instances.

Table 2: The results from the computational experiments, the second setting. Geometric average of runtimes in seconds over 10 independent runs.

			batch size										
n	m	WA	10	20	50	100	200	500	1,000	2,000	5,000	10,000	speedup
5	1,000	0.002	0.003	0.002	0.002	0.003	0.003	0.003	-	-	-	-	0.730
	5,000	0.003	0.006	0.008	0.005	0.005	0.005	0.005	0.006	0.007	-	-	0.761
	10,000	0.002	0.003	0.003	0.003	0.003	0.003	0.003	0.004	0.005	0.009	-	0.566
	50,000	0.006	0.010	0.010	0.007	0.007	0.007	0.008	0.009	0.010	0.017	0.026	0.833
	100,000	0.011	0.017	0.017	0.011	0.012	0.012	0.012	0.013	0.015	0.024	0.035	0.963
	500,000	0.058	0.057	0.036	0.037	0.037	0.038	0.038	0.041	0.043	0.051	0.063	1.620
	1,000,000	0.117	0.109	0.111	0.074	0.072	0.074	0.075	0.077	0.081	0.092	0.108	1.617
	5,000,000	0.533	0.471	0.313	0.314	0.313	0.313	0.316	0.316	0.320	0.334	0.359	1.702
10	1,000	0.002	0.007	0.007	0.004	0.005	0.005	0.005	-	-	-	-	0.558
	5,000	0.002	0.006	0.006	0.003	0.003	0.004	0.004	0.005	0.006	-	-	0.594
	10,000	0.003	0.008	0.006	0.008	0.008	0.004	0.005	0.006	0.007	0.014	-	0.682
	50,000	0.012	0.024	0.017	0.015	0.010	0.011	0.012	0.012	0.014	0.022	0.033	1.179
	100,000	0.021	0.036	0.026	0.026	0.018	0.018	0.019	0.020	0.022	0.031	0.043	1.185
	500,000	0.110	0.098	0.095	0.064	0.066	0.066	0.068	0.069	0.073	0.085	0.101	1.711
	1,000,000	0.208	0.240	0.179	0.181	0.122	0.121	0.123	0.126	0.128	0.141	0.160	1.720
	5,000,000	0.998	0.852	0.851	0.581	0.592	0.590	0.587	0.591	0.590	0.614	0.630	1.717
20	1,000	0.004	0.014	0.008	0.009	0.008	0.006	0.007	-	-	-	-	0.638
	5,000	0.005	0.017	0.015	0.011	0.008	0.008	0.008	0.009	0.011	-	-	0.701
	10,000	0.008	0.024	0.019	0.013	0.009	0.009	0.010	0.011	0.012	0.020	-	0.900
	50,000	0.026	0.051	0.037	0.029	0.020	0.020	0.021	0.022	0.024	0.034	0.049	1.306
	100,000	0.050	0.081	0.063	0.048	0.032	0.033	0.035	0.036	0.039	0.048	0.064	1.533
	500,000	0.239	0.316	0.252	0.253	0.190	0.127	0.128	0.131	0.133	0.146	0.166	1.882
	1,000,000	0.458	0.483	0.359	0.243	0.240	0.242	0.244	0.243	0.248	0.261	0.283	1.912
	5,000,000	2.246	2.847	2.322	1.742	1.164	1.163	1.172	1.163	1.171	1.197	1.221	1.932
50	1,000	0.015	0.084	0.048	0.031	0.037	0.034	0.022	-	-	-	-	0.716
	5,000	0.024	0.113	0.080	0.048	0.041	0.042	0.029	0.029	0.031	-	-	0.849
	10,000	0.030	0.140	0.070	0.055	0.044	0.045	0.045	0.033	0.035	0.048	-	0.926
	50,000	0.133	0.251	0.167	0.110	0.084	0.055	0.056	0.057	0.061	0.079	0.102	2.429
	100,000	0.248	0.416	0.284	0.188	0.144	0.094	0.096	0.099	0.099	0.118	0.143	2.634
	500,000	1.211	1.680	1.133	0.935	0.568	0.564	0.566	0.383	0.393	0.409	0.438	3.163
	1,000,000	2.329	3.237	2.171	1.446	1.460	1.087	0.728	0.737	0.739	0.753	0.779	3.199
	5,000,000	11.293	13.620	10.249	6.828	5.059	3.423	3.429	3.466	3.526	3.485	3.499	3.299
100	1,000	0.060	0.552	0.294	0.156	0.117	0.163	0.078	-	-	-	-	0.770
	5,000	0.077	0.712	0.380	0.225	0.183	0.144	0.153	0.099	0.113	-	-	0.774
	10,000	0.141	0.699	0.412	0.248	0.193	0.160	0.101	0.107	0.117	0.144	-	1.402
	50,000	0.500	1.373	0.762	0.421	0.344	0.355	0.277	0.284	0.197	0.222	0.295	2.533
	100,000	0.927	2.150	1.153	0.653	0.529	0.401	0.411	0.279	0.283	0.315	0.387	3.321
	500,000	4.485	7.458	4.177	2.320	1.858	1.881	0.951	0.945	0.974	0.976	1.036	4.747
	1,000,000	8.542	14.941	8.006	4.382	3.566	2.643	2.658	1.783	1.795	1.769	1.887	4.828
	5,000,000	41.466	61.607	34.675	19.387	15.415	11.481	11.730	11.725	7.947	7.850	8.077	5.282
200	1,000	0.253	3.953	2.111	0.996	0.667	0.531	0.517	-	-	-	-	0.490
	5,000	0.438	5.303	2.978	1.496	0.919	0.552	0.589	0.398	0.435	-	-	1.099
	10,000	0.755	5.705	3.098	1.569	1.018	0.581	0.643	0.648	0.647	0.690	-	1.298
	50,000	1.913	8.707	4.651	2.346	1.510	1.197	0.958	1.000	1.028	1.247	1.206	1.996
	100,000	3.281	12.114	6.737	3.284	2.050	1.710	1.295	0.865	0.915	1.184	1.519	3.796
	500,000	14.543	42.987	21.998	10.566	6.537	5.376	3.999	4.046	4.097	3.086	3.337	4.713
	1,000,000	27.823	74.570	38.847	19.303	12.151	9.568	7.290	7.508	4.911	5.278	5.569	5.666
	5,000,000	135.670	352.154	187.454	89.096	55.879	44.527	33.654	33.772	33.807	22.997	23.269	5.900

The results for the first experimental setting show that the Algorithm 3 was more than 30 times faster for the largest instances (the speedup column in the tables is the ratio between the runtime of the WA algorithm and the Algorithm 3 with "optimally" chosen batch size). It also showed, that at least for the first setting, the "optimal" batch size (the k, for which the algorithm converged the fastest) quite surprisingly did not appear to depend on the dimension n of the MVCE problem.

The results from the second experiment setting paint a rather different picture. The speedup, although still significant, was much smaller when compared to the first setting. Moreover, the "optimal" batch size k quite clearly depends on the dimension of the MVCE problem. An important observation is that there is no clear dependence on the number of vectors m. A reasonable rule of thumb for the choice of k for problems that are similar to the second setting seems to be between $k = 5n$ and $k = 20n$.

5 Conclusion

The MVCE problem comes up in various real-world problems in engineering and statistics. In this paper, we described an improving pooling and batching scheme for the Wolfe-Atwood method for large-scale MVCE instances. We conducted quite extensive computational experiments and showed that the "optimal" batch size k depends on the structure of the problem – there is a qualitative difference in its dependence on the dimension n of the problem between first and second experimental setting. The computational experiments showed great promise, regarding the implementability and the usefulness of the proposed algorithm, but there is still much work to be done. The proposed method should be implemented on real-life problems, to get a clearer picture of the improvements it provides. The importance of the selection of the starting set $\tilde{\mathcal{X}}$ and different selection criteria for updating said set should be further looked into. Another possible line of research constitutes the inclusion of the discarding methodology developed in [21] for "Ellipsoidal peeling" in experiment design [34].

Acknowledgement: This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic, INTER-COST project LTC18053 and European COST Action CA15140 and by IGA BUT: No. FSI-S-17-4785..

References

[1] Todd, M. J. 2016. *Minimum-Volume Ellipsoids: Theory and Algorithms*. SIAM, Philadelphia, USA. ISBN 9781611974379.

[2] Danzer, L., Laugwitz, D., and Lenz, H. 1957. Über das Löwnersche ellipsoid und sein analogon unter den einem eikörper einbeschreibenen ellipsoiden. *Archiv der Mathematik* 8,l pp. 214–219.

[3] Zaguskin, V. L. 1958. Circumscribed and inscribed ellipsoids of extremal volume (in Russian). *Uspehi Matematicheskikh Nauk* 13, pp. 89–93.

[4] John, F. 1948. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays, presented to R. Courant on his 60th birthday*, Interscience, New York, pp. 187–204.

[5] Silvey, S. D. 1980. *Optimal Design: An Introduction to the Theory for Parameter Estimation.* Chapman and Hall, New York.

[6] Silverman, B. W. and Titterington, D. M. 1980. Minimum covering ellipses. *SIAM Journal on Scientific and Statistical Computing* 1, pp. 401–409.

[7] Holesovsky, J. and Kudela, J. 2016. Outlier identification based on local extreme quantile estimation. *Mendel*, 22, 1, pp. 255–260.

[8] Somplak, R., Smidova, Z., Smejkalova, V., and Nevrly, N. 2018. Statistical evaluation of large-scale data logistics system. *Mendel*, 24, 2, pp. 9–16. DOI: 10.13164/mendel.2018.2.009

[9] Gill, P. E., Golub, G. H., Murray, W. , and Saunders, M. A. 1974. Methods for modifying matrix factorizations. *Mathematics of Computation* 28, 505–535.

[10] Viktorin, A., Senkerik, R., Pluhacek, M., and Kadavy T. 2018. Clustering analysis of the population in Db_SHADE algorithm. *Mendel*, 24, 1, pp. 9–16. DOI:10.13164/mendel.2018.1.009

[11] Grabusts, P. 2018. Numerical data clustering ontology approach. *Mendel*, 24, 1, pp. 31–38. doi:10.13164/mendel.2018.1.031

[12] Hrabec, D., Kudela, J., Somplak, R., Nevrly, V., and Popela, P. 2019. Circular economy implementation in waste management network design problem: a case study. Preprint in *Central European Journal of Operations Research*. DOI: 10.1007/s10100-019-00626-z

[13] Chernousko, F. L. 2005. Ellipsoidal state estimation for dynamical systems. *Nonlinear Analysis* 63, pp. 872–879.

[14] Eberly, D. 2001. *3D Game Engine Design*. Morgan Kaufmann, San Francisco, CA.

[15] Lacoste-Julien, S. and Jaggi, M. 2015. On the global linear convergence of Frank-Wolfe optimization variants. In *C. Cortes et al., editors, Advances in Neural Information Processing Systems 28 (NIPS 2015)*, Curran Associates, Inc., New York, pp. 496–504.

[16] Pena, J. and Rodriguez, D. 2015. *Polytope conditioning and linear convergence of the Frank–Wolfe algorithm.* Technical report. Tepper School of Business, Carnegie-Mellon University, Pittsburgh, PA.

[17] Wolfe, P. 1970. Convergence theory in nonlinear programming. In *J. Abadie, editor, Integer and Nonlinear Programming*, North-Holland, Amsterdam, pp. 1–36.

[18] Atwood, C. L. 1973. Sequences converging to D-optimal designs of experiments. *The Annals of Statistics* 1, pp. 342–352.

[19] Kumar, P. and Yıldırım, E. A. 2005. Minimum-volume enclosing ellipsoids and core sets. *J. Optimization Theory and Applications* 126, 1, pp. 1–21.

[20] Kudela, J. and Popela, P. 2018. Chance constrained optimal beam design: Convex reformulation and probabilistic robust design. *Kybernetika* 54, 6, pp. 1201–1217.

[21] Kudela, J. 2019. *Advanced Decomposition Methods in Stochastic Convex Optimization*. Doctoral thesis, Brno University of Technology, Brno, Czech Republic.
[22] Todd, M. J. and Yıldırım, E. A. 2007. On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids. *Discrete and Applied Mathematics* 155, 13, pp. 1731–1744.
[23] Boyd, S. and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.
[24] Frank, M. and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly* 3, pp. 95–110.
[25] Fedorov, V. V. 1972. *Theory of Optimal Experiments*. Academic Press, New York.
[26] Wynn, H. P. 1970. The sequential generation of D-optimum experimental design. *Annals of Mathematical Statistics* 41, pp. 1655–1664.
[27] Ahipasaoglu, S. D., Sun, P., and Todd, M. J. 2008. Linear convergence of a modified Frank–Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software* 23, pp. 5–19.
[28] Sun, P. and Freund, R. M. 2004. Computation of minimum volume covering ellipsoids. *Operations Research* 52, pp. 690–706.
[29] Khachiyan, L. G. 1996. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research* 21, 307–320.
[30] Betke, U. and Henk, M. 1993. Approximating the volume of convex bodies. *Discrete Computational Geometry* 10, 15–21.
[31] Harman, R. and Pronzato, L. 2007. Improvements on removing nonoptimal support points in D-optimum design algorithms. *Statistics and Probability Letters* 77, pp. 90–94.
[32] Ahipasaoglu, S. D. 2009. *Solving ellipsoidal inclusion and optimal experimental design problems: Theory and algorithms*. Doctoral thesis, Cornell University, Ithaca, NY.
[33] Kudela, J. and Popela, P. 2017. Warm-start cuts for Generalized Benders Decomposition. *Kybernetika* 53, 6, pp. 1012–1025. DOI:10.14736/kyb-2017-6-1012
[34] Ahipasaoglu, S. D. 2015. Fast algorithms for the minimum volume estimator. *J. Global Optimization* 62, pp. 351–370. DOI: 10.1007/s10898-014-0233-8