Computing *D*-optimal approximate designs of experiments on finite spaces: a survey and comparison of algorithms

Radoslav Harman

Department of Applied Mathematics and Statistics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
Slovakia

Design of Experiments: New Challenges
CIRM Marseilles 2018

Overview of the talk

- **Notation and problem statement**
- **Survey of algorithms** (Vertex-direction, Away-step, Multiplicative, Vertex-exchange, Cocktail, Other algorithms)
- **A randomized batch-exchange method**
- **Numerical comparison of the *D*-optimal design algorithms**

Partially based on the manuscript:

Radoslav Harman, Lenka Filová, Peter Richtárik: *A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments*, arXiv:1801.05661 [stat.CO]

Notation and problem statement

Our aim is to compute a solution $\mathbf{w}^*$ to the problem

$$\max \det \left(\sum_{i=1}^{n} w_i \mathbf{x}_i \mathbf{x}_i^T \right), \quad \text{subject to } \mathbf{w} \in \mathcal{W}, \tag{1}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ are known vectors, and $\mathcal{W}$ is the simplex:

$$\mathcal{W} = \{\mathbf{w} \in [0,1]^n : \mathbf{1}_n^T \mathbf{w} = 1\}.$$

In experimental design, $\mathbf{w} \in \mathcal{W}$ represents an "approximate design" of a regression model with m parameters and "regressors" (or vectors of "covariates") $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Any solution $\mathbf{w}^*$ of (1) is a "*D*-optimal approximate design".

The problem (1) with $\mathbf{x}_i = (1, \mathbf{y}_i^T)^T$, $i = 1, \ldots, n$, is equivalent to the problem of finding the minimum-volume ellipsoid enclosing the data $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^{m-1}$. The MVEE has numerous applications beyond design of experiments.

Notation and problem statement

The "information matrix" mapping $\mathbf{M} : \mathbb{R}^m \to \mathcal{S}^m$ defined by

$$\mathbf{M}(\mathbf{w}) = \sum_{i=1}^{n} w_i \mathbf{x}_i \mathbf{x}_i^T$$

is linear on $\mathbb{R}^m$, and the "criterion of *D*-optimality" $\Phi : \mathcal{S}_+^m \to [0, \infty)$

$$\Phi(\mathbf{M}) = (\det(\mathbf{M}))^{1/m}$$

is concave on $\mathcal{S}_+^m$. Therefore, (1) is a problem of convex optimization.

Because $\mathbf{M}(\mathcal{W}) \subset \mathcal{S}_+^m$ is compact and Φ is continuous on $\mathcal{S}_+^m$, there is always at least one *D*-optimal design $\mathbf{w}^*$. Moreover, $\mathbf{M}(\mathbf{w}^*)$ is a unique non-singular matrix (if $\operatorname{span}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \mathbb{R}^m$).

Sometimes we are able to find the analytic form of a *D*-optimal design; usually, however, we must use a computer to find a *D*-optimal design numerically.

Algorithms for *D*-optimal design (in general)

1 Generate $\mathbf{w}_1 \in \mathcal{W}$ such that $\mathbf{M}(\mathbf{w}_1)$ is non-singular; set $k \leftarrow 1$
2 **while** $\mathbf{w}_k$ *does not satisfy a stopping rule* **do**
3 | Compute $\mathbf{w}_{k+1} \in \mathcal{W}$ based on $\mathbf{w}_k$
4 | Set $k \leftarrow k + 1$
5 **end**

An algorithm "converges to *D*-optimality" if (with no stopping rule):

$$\lim_{k \to \infty} \Phi(\mathbf{M}(\mathbf{w}_k)) = \Phi(\mathbf{M}(\mathbf{w}^*)),$$

where $\mathbf{w}^*$ is any *D*-optimal design.

We will not discuss the methods of generating an initial design nor stopping rules; we will focus on Step 3 of algorithms with proved convergence to *D*-optimality.

Note: There is a large class of powerful "population" methods such as genetic algorithms and particle swarm optimization (Wong et al), which store and improve an entire *set* of feasible designs. However, the real power of these methods is in non-convex optimization.

Algorithms for *D*-optimal design (in general)

For $\mathbf{w} \in \mathcal{W}$ such that $\mathbf{M}(\mathbf{w}) \in \mathcal{S}^m_{++}$ define the "variance function" by

$$d(\mathbf{x}_i, \mathbf{w}) := \mathbf{x}_i^T \mathbf{M}^{-1}(\mathbf{w}) \mathbf{x}_i, \; i = 1, \ldots, n.$$

Interpretation of $d(\mathbf{x}_i, \mathbf{w})$: Variance of the BLUE of the response expectation in $\mathbf{x}_i$, a directional derivative of a version of criterion of *D*-optimality, the leveraging coefficient h_{ii} ($\rightarrow$ the talk of J.Stufken)...

It is simple to show that for any $\mathbf{w}^* \in \mathcal{W}$ such that $\mathbf{M}(\mathbf{w}^*) \in \mathcal{S}^m_{++}$:

$$\begin{aligned} &\mathbf{w}^* \text{ is } D\text{-optimal} \Leftrightarrow \\ &\mathbf{w}^* \text{ minimizes } \max_i d(\mathbf{x}_i, \mathbf{w}) \Leftrightarrow \\ &\max_i d(\mathbf{x}_i, \mathbf{w}^*) = m. \end{aligned}$$

The first equivalence above is the original "equivalence theorem" between *D*- and *G*-optimality (Kiefer & Wolfowitz 1969).

Most algorithms for *D*-optimality are inspired by this equivalence in the sense that they try to minimize the maximum of d by increasing w_i for one of more regressors $\mathbf{x}_i$ with a large value of $d(\mathbf{x}_i, \mathbf{w})$.

Recent algorithms also utilize the sparsity of $\mathrm{supp}(\mathbf{w}^*) = \{i : w_i^* > 0\}$.

Vertex direction method (VDM)

Fedorov 1972; related to the metod of Frank & Wolfe 1956.

1 **repeat**
2 | Find $i^* \in \operatorname{argmax}_i d(\mathbf{x}_i, \mathbf{w})$
3 | Find $\alpha^* \in \operatorname{argmax}_\alpha \Phi[\, \mathbf{M}(\alpha \mathbf{w} + (1-\alpha)\mathbf{e}_{i^*}) \,]$
4 | Set $\mathbf{w} \leftarrow \alpha^* \mathbf{w} + (1-\alpha^*)\mathbf{e}_{i^*}$
5 **end**;

There is an explicit formula for α^* in Step 2.
Variants: a different selection of α^* in Step 2 (e.g., Wynn 1970).

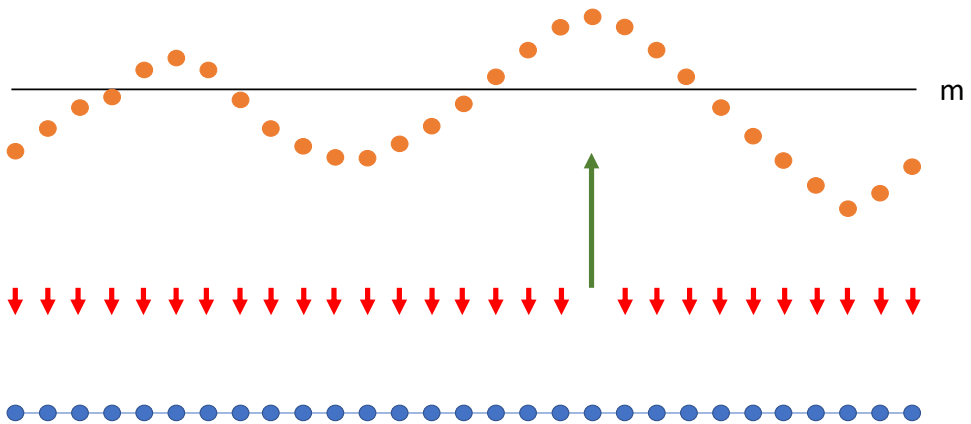

Figure: Design points, Variance function, Decrease of *w*, Increase of *w*

Vertex direction method (VDM)

Aspect	Evaluation
Simplicity of the idea and coding	Very good
Speed of computation	Bad
Possibility to include non-standard design constraints	Good*
Possibility to use for other criteria	Very good

*Cook & Fedorov 1995

Away-steps method (AWS)

Atwood 1973, which is realted to a previous work of Wolfe 1970

repeat
  Find $k \in \operatorname{argmin}\{d(\mathbf{x}_u, \mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\}$
  Find $l \in \operatorname{argmax}\{d(\mathbf{x}_v, \mathbf{w}) : v \in \{1, \ldots, n\}\}$
  if $\frac{1}{2}[d(\mathbf{x}_k, \mathbf{w}) + d(\mathbf{x}_l, \mathbf{w})] > m$ **then**
    Find a smart $\alpha^* \in [0, 1]$
    $\mathbf{w} \leftarrow \alpha^* \mathbf{w} + (1 - \alpha^*)\mathbf{e}_l$
  end
  else
    Find a smart $\alpha^* \in [1, 1 + w_k]$
    $\mathbf{w} \leftarrow \alpha^* \mathbf{w} + (1 - \alpha^*)\mathbf{e}_k$
  end
end;

Used in, e.g., Todd 2016.

Away-steps method (AWS)

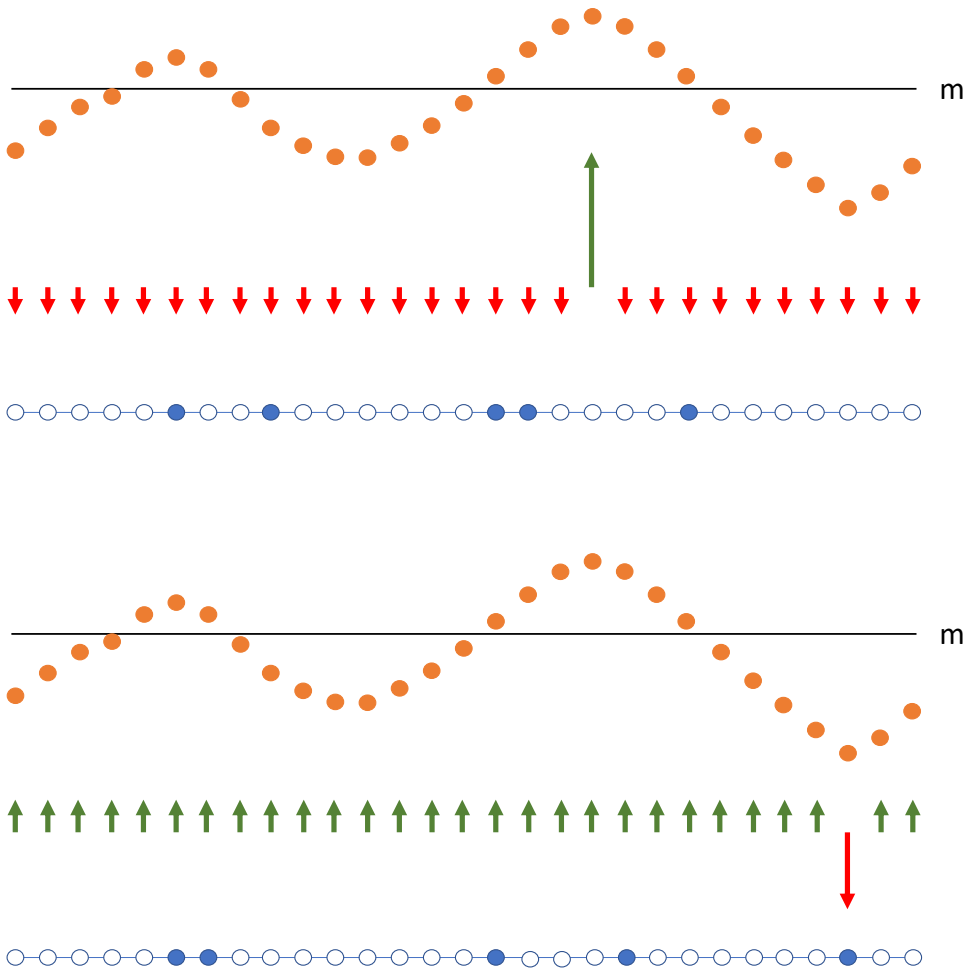

Figure: Design points, Variance function, Decrease of w, Increase of w

Away-step method (AWS)

Aspect	Evaluation
Simplicity of the idea and coding	Good
Speed of computation	Good
Possibility to include non-standard design constraints	I do not know
Possibility to use for other criteria	Good

Multiplicative method (MUL)

Torsney, Titterigton & Silvey 1970's

1 **repeat**
2 | Set $\mathbf{d}(\mathbf{w}) \leftarrow (d(\mathbf{x}_1, \mathbf{w}), \ldots, d(\mathbf{x}_n, \mathbf{w}))^T$
3 | Set $\mathbf{w} \leftarrow m^{-1}\mathbf{d}(\mathbf{w}) \odot \mathbf{w}$
4 **end**;

Variants: for instance using a "longer" Step 3 (e.g., Dette, Pepelyshev & Zhigljavsky 2008)

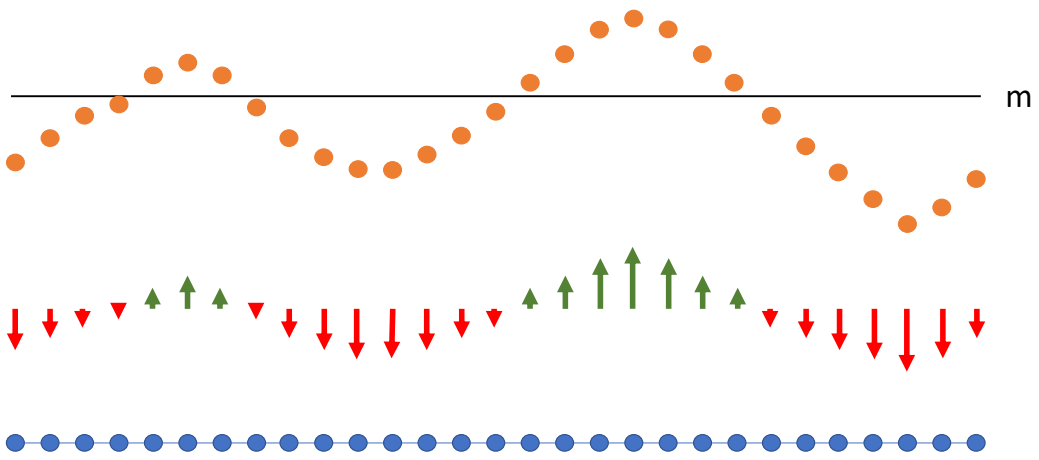

Figure: Design points, Variance function, Decrease of w, Increase of w

Multiplicative method (MUL)

Aspect	Evaluation
Simplicity of the idea and coding	Very good
Speed of computation	Good
Possibility to include non-standard design constraints	Problematic*
Possibility to use for other criteria	Good**

*For simple constraints see Harman 2014, Harman & Benková 2017
**Heuristically, we can use MUL with almost any criterion, but convergence proved only for some criteria; see Yu 2010

Vertex exchange algorithm (VEM)

Böhning 1986

1 **repeat**
2 | Find $k \in \operatorname{argmin}\{d(\mathbf{x}_u, \mathbf{w}) : u \in \operatorname{supp}(\mathbf{w})\}$
3 | Find $l \in \operatorname{argmax}\{d(\mathbf{x}_v, \mathbf{w}) : v \in \{1, \ldots, n\}\}$
4 | Find α^* belong to $\operatorname{argmax}\{\Phi_D[\,\mathbf{M}(\mathbf{w} + \alpha\mathbf{e}_l - \alpha\mathbf{e}_k)\,] : \alpha \in [-w_l, w_k]\}$
5 | $w_k \leftarrow w_k - \alpha^*$; $w_l \leftarrow w_l + \alpha^*$
6 **end**;

There is an explicit formula for α^* in Step 4.

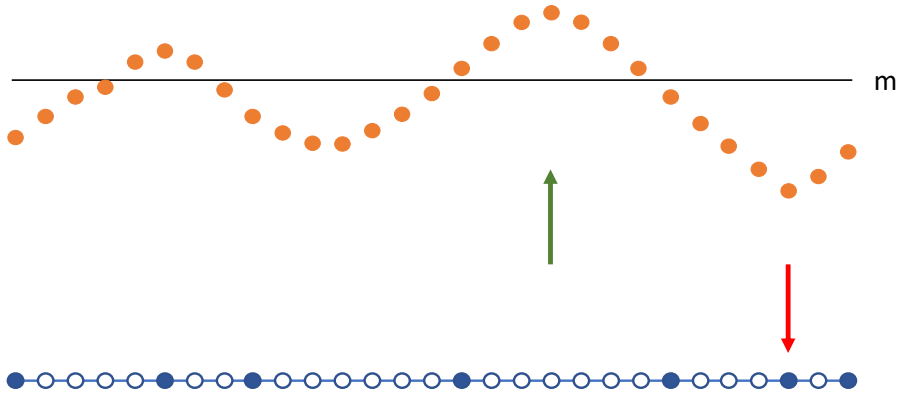

Figure: Design points, Variance function, Decrease of w, Increase of w

Vertex exchange algorithm (VEX)

Aspect	Evaluation
Simplicity of the idea and coding	Good
Speed of computation	Often good
Possibility to include non-standard design constraints	I do not know
Possibility to use for other criteria	Possible*

*But I am not aware of any convergence result for criteria other than *D*-optimality.

Cocktail algorithm (COA)

Yu 2011

A clever hybrid of the methods VDM, MUL and VEX, where the vertex exchange is performed on "neighbouring" design points (we will not formulate it as a pseudo-code).

Aspect	Evaluation
Simplicity of the idea and coding	Average
Speed of computation	Very good
Possibility to include non-standard design constraints	I do not know
Possibility to use for other criteria	Possible*

*But I am not aware of any convergence result for criteria other than D-optimality.

Other algorithms for *D*-optimality

Simplicial decomposition (SIM; Ucinski & Patan 2007 in design of experiments, but the idea dates back to 70s): In each iteration, a simple sub-problem is solved, which generates an extreme point of the polyhedron of feasible solutions, and a non-linear restricted master problem finds the maximum of the objective function over the convex hull (a simplex) of previously defined extreme points.

A very efficient variant of SIM for computing optimal designs has been proposed in Yang, Biedermann & Tang 2013, where the master problem is solved by a Newton method.

Semidefinite and second-order programming methods can also be used for computing *D*-optimal designs (e.g., Vandenberghe, Boyd & Wu 1998, Sagnol & Harman 2015). A major disadvantage is that these algorithms can be used only for small size of the problems (n in the order of thousands with standard hardware), although it has huge other advantages (e.g., possibility to include general linear constraints $\rightarrow$ the talk of F. Gamboa).

Randomized batch-exchange algorithm (REX)

Harman, Filová & Richtárik 2018

1 **repeat**
2 | Perform steps 2-5 of VEX.
3 | $\mathbf{k} \leftarrow$ a random permutation of elements of $\mathrm{supp}(\mathbf{w})$
4 | $\mathbf{l} \leftarrow$ a random permutation of the indices of L greatest elements of $d(\mathbf{x}_i, \mathbf{w})$, $i = 1, \ldots, n$
5 | **for** *l in* $1 : L$ **do**
6 | | **for** *k in* $1 : \#\mathrm{supp}(\mathbf{w})$ **do**
7 | | | Find $\alpha^* \in \mathrm{argmax}\{\Phi[\,\mathbf{M}(\mathbf{w} + \alpha(\mathbf{e}_{\mathbf{l}_l} - \mathbf{e}_{\mathbf{k}_k}))\,] : \alpha \in [-w_{\mathbf{l}_l}, w_{\mathbf{k}_k}]\}$
8 | | | **if** *Step 2 was not nullifying or* $\alpha^* = -w_{\mathbf{l}_l}$ *or* $\alpha^* = w_{\mathbf{k}_k}$ **then**
9 | | | | $w_{\mathbf{k}_k} \leftarrow w_{\mathbf{k}_k} - \alpha^*$ and $w_{\mathbf{l}_l} \leftarrow w_{\mathbf{l}_l} + \alpha^*$
10 | | | **end**
11 | | **end**
12 | **end**
13 **end**;

L can be chosen as a constant or based on some heuristic rule. Step 2 is "nullifying", if it nullified some of the weights that it exchanged. There is an explicit formula for α^* in Step 7.

Randomized exchange algorithm (REX)

REX can be viewed as analogous the KL exchange algorithm for computing *exact D*-optimal designs (Atkinson et al. 2007), but instead of finding the best exchange between K times L pairs of points, we distribute the weights optimally between the pairs of design points based on very fast formulas. (And we do it in a random order.)

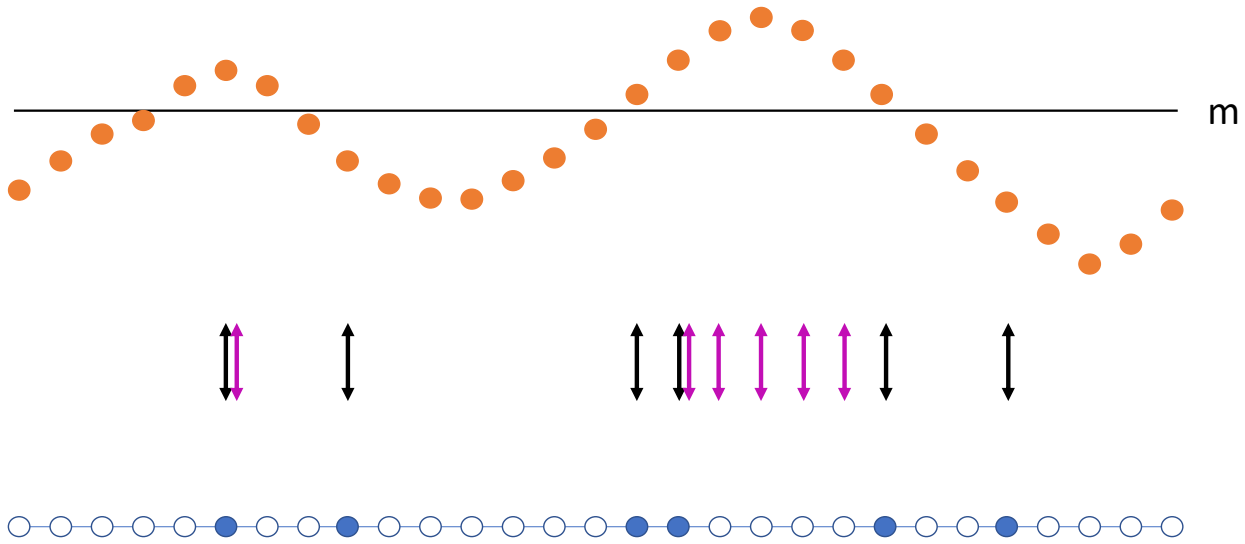

Figure: Design points, Variance function, Support points weights change, Maximum variance points weights change

Randomized batch-exchange algorithm (REX)

Aspect	Evaluation
Simplicity of the idea and coding	Average
Speed of computation	Excellent
Possibility to include non-standard design constraints	I do not know
Possibility to use for other criteria	Possible*

*We derived optimal-exchange formulas for *A*-optimality, but were unable to prove the convergence of REX for *A*-optimality, despite strong numerical evidence.

However, we were able to prove convergence of REX for the criterion of *D*-optimality (see the manuscript for a rather long and technical proof).

Numerical comparison

Important factors that influence the computation are n, m, but also the position of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ themselves (the "structure" of the problem).

We selected various combinations of n, m and generated $\mathbf{x}_1, \ldots, \mathbf{x}_n$ independently randomly from $N_m(\mathbf{0}_m, \mathbf{I}_m)$ to imitate a "general" problem of D-optimality.

Of course, each specific selection of the regressors leads to different performance of the algorithms, but we have also tested several other situations (e.g., the regressors of polynomial models with one or several explanatory variables), and the numerical results are similar.

As for the implementation, we used author-supplied R codes for COA and did our best to convert the other competing algorithms to R.

In the following graphs, "log-efficiency of 1" means efficiency 0.9, "log-efficiency of 2" means efficiency of 0.99, etc. All the efficiencies are relative to a pre-computed, practically perfectly D-optimal design.

Numerical comparison

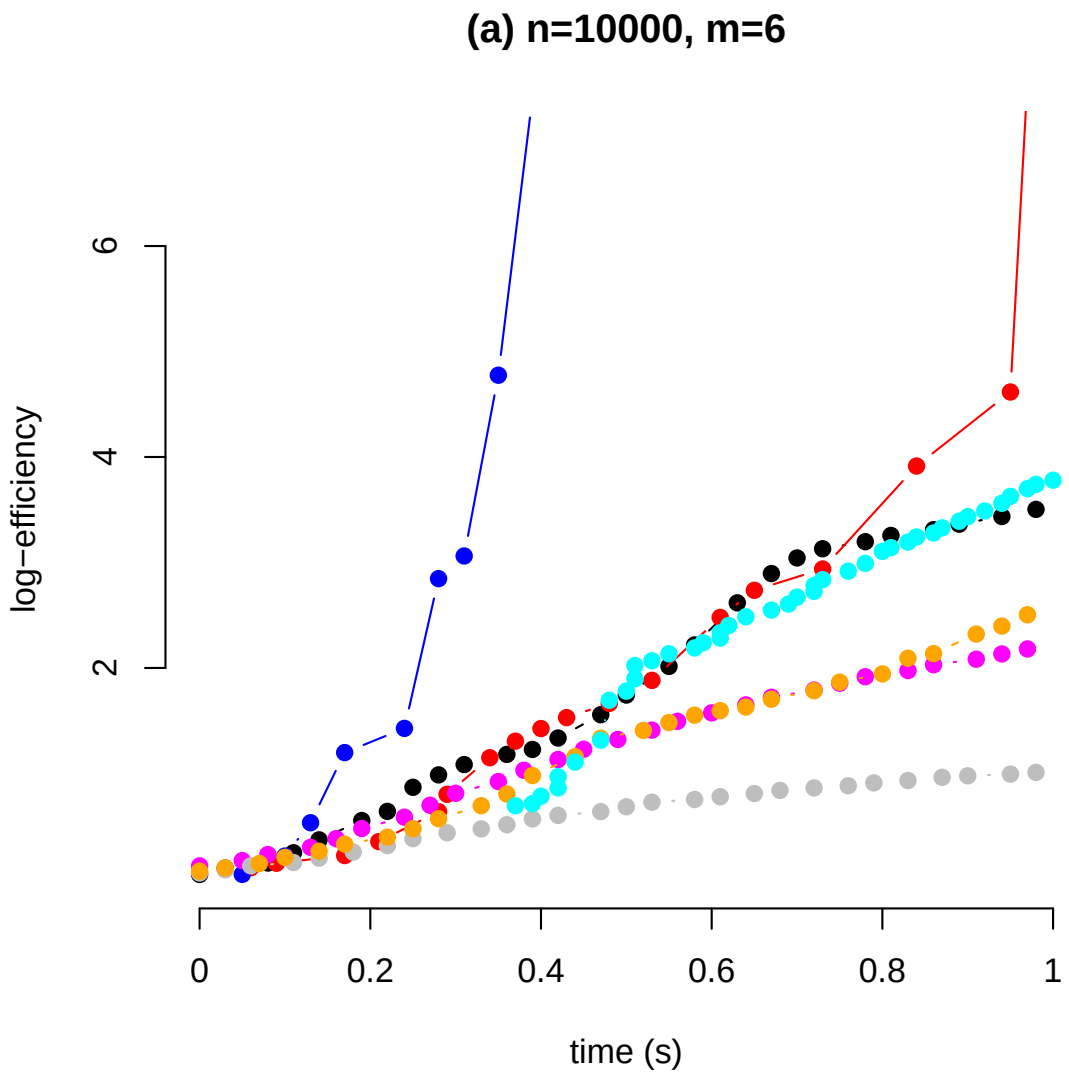

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Numerical comparison

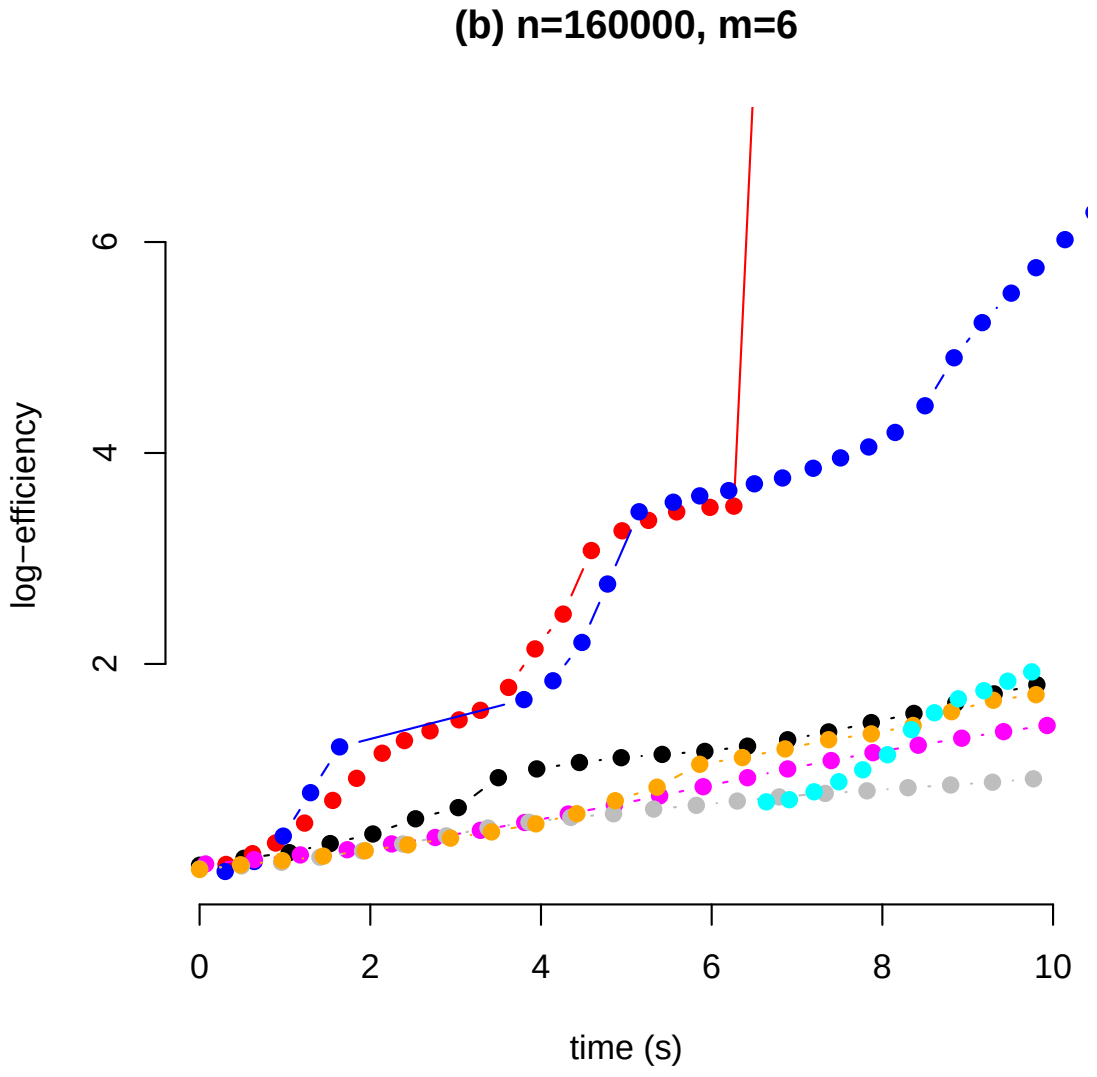

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Numerical comparison

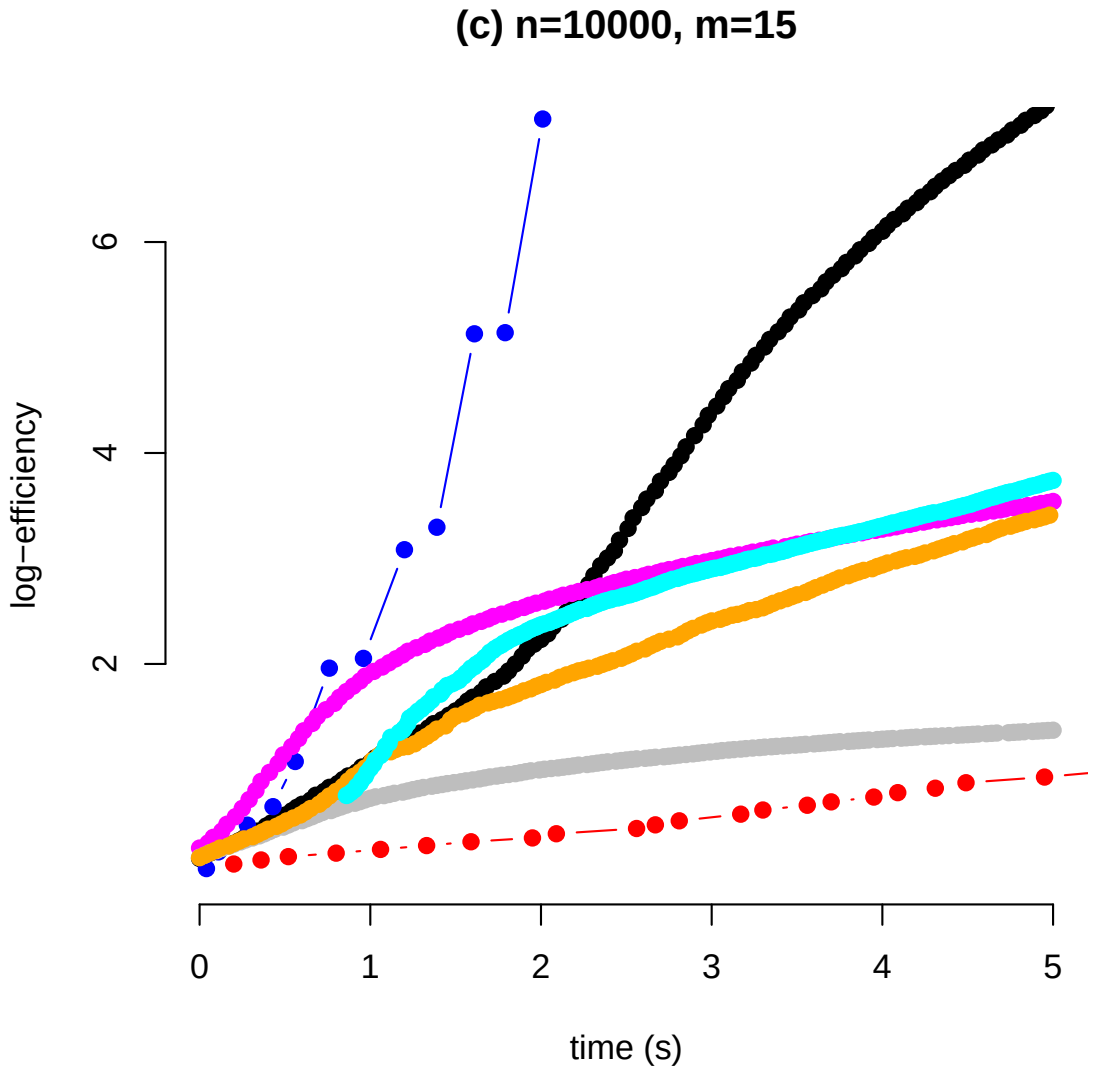

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Numerical comparison

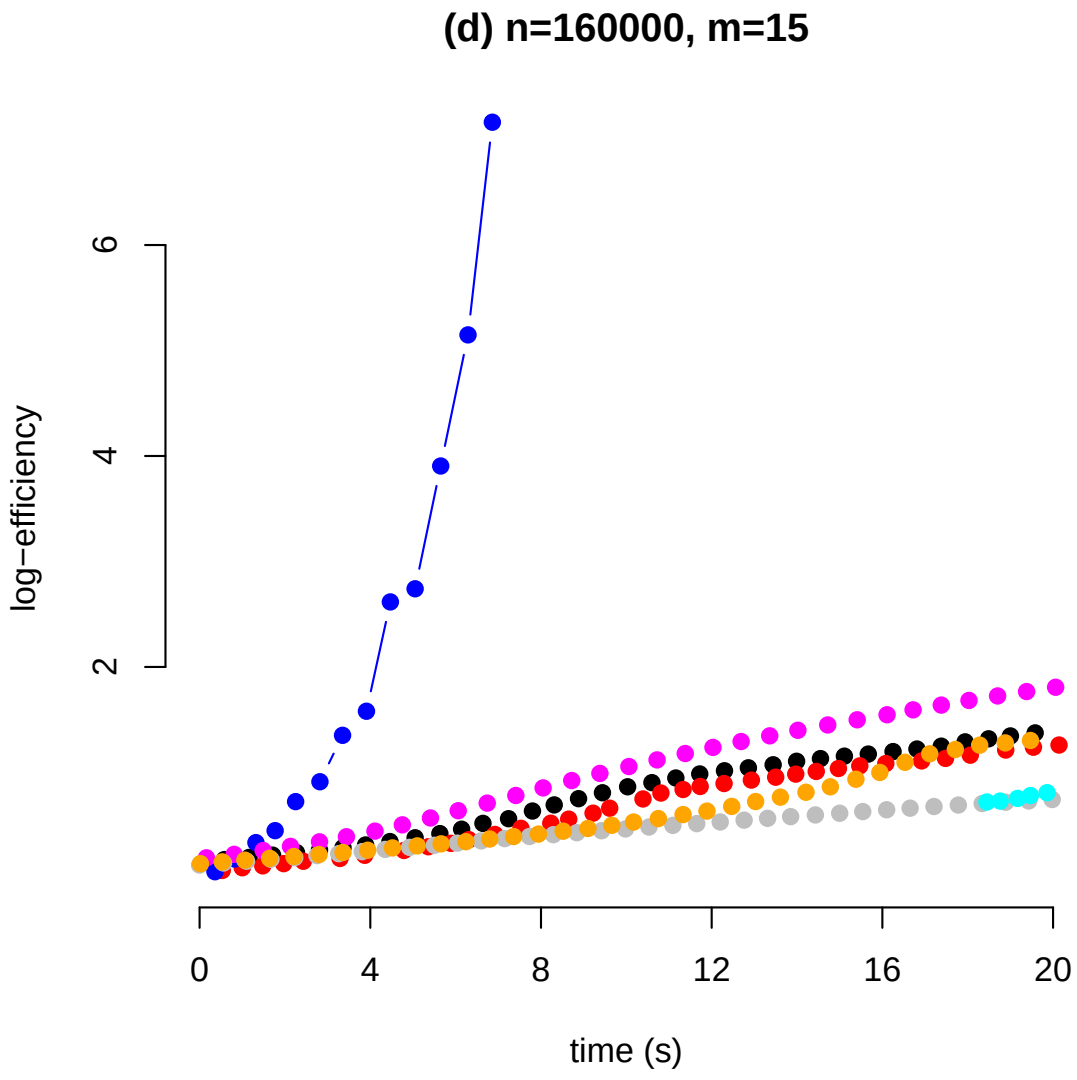

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Numerical comparison

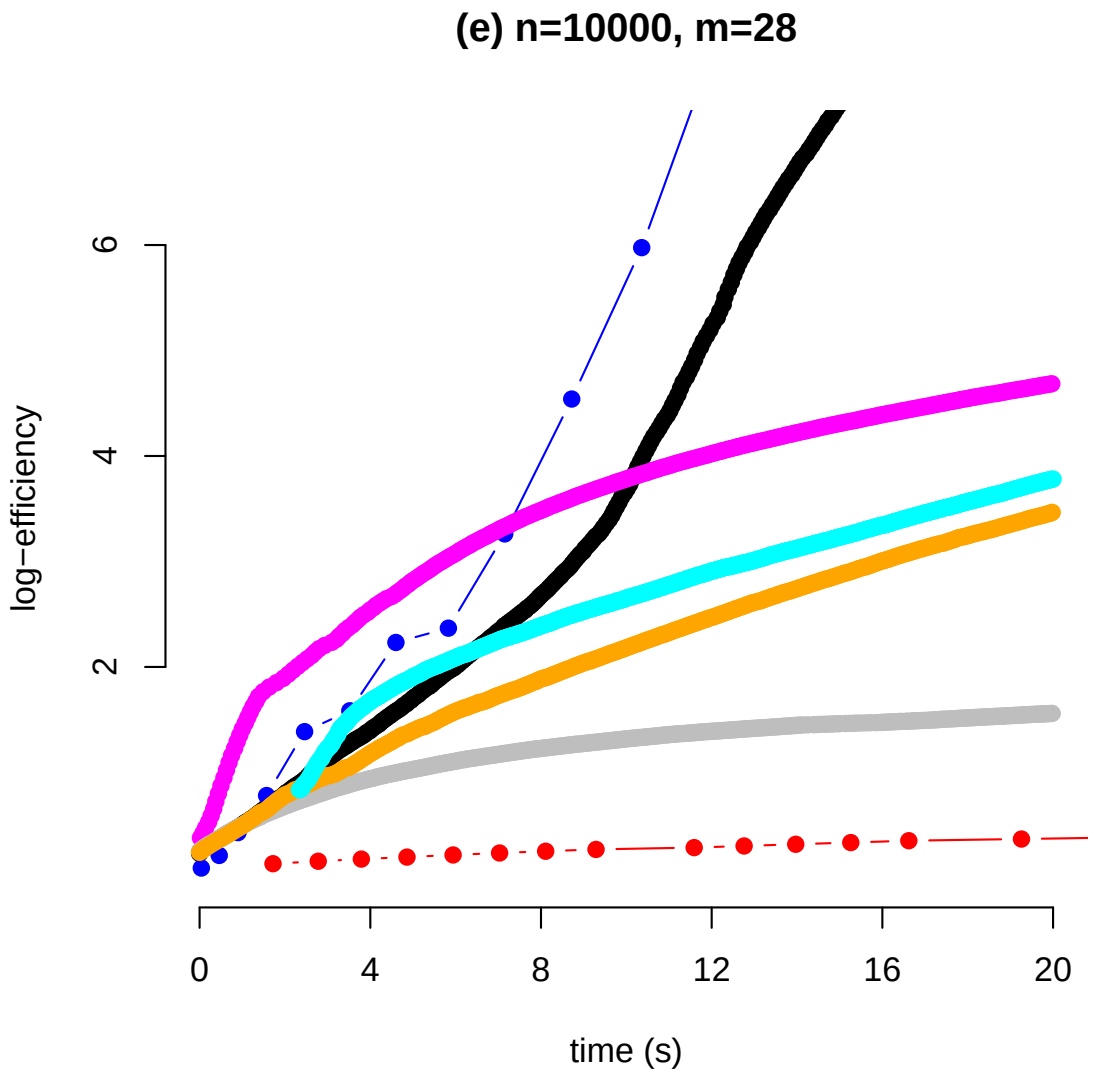

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Numerical comparison

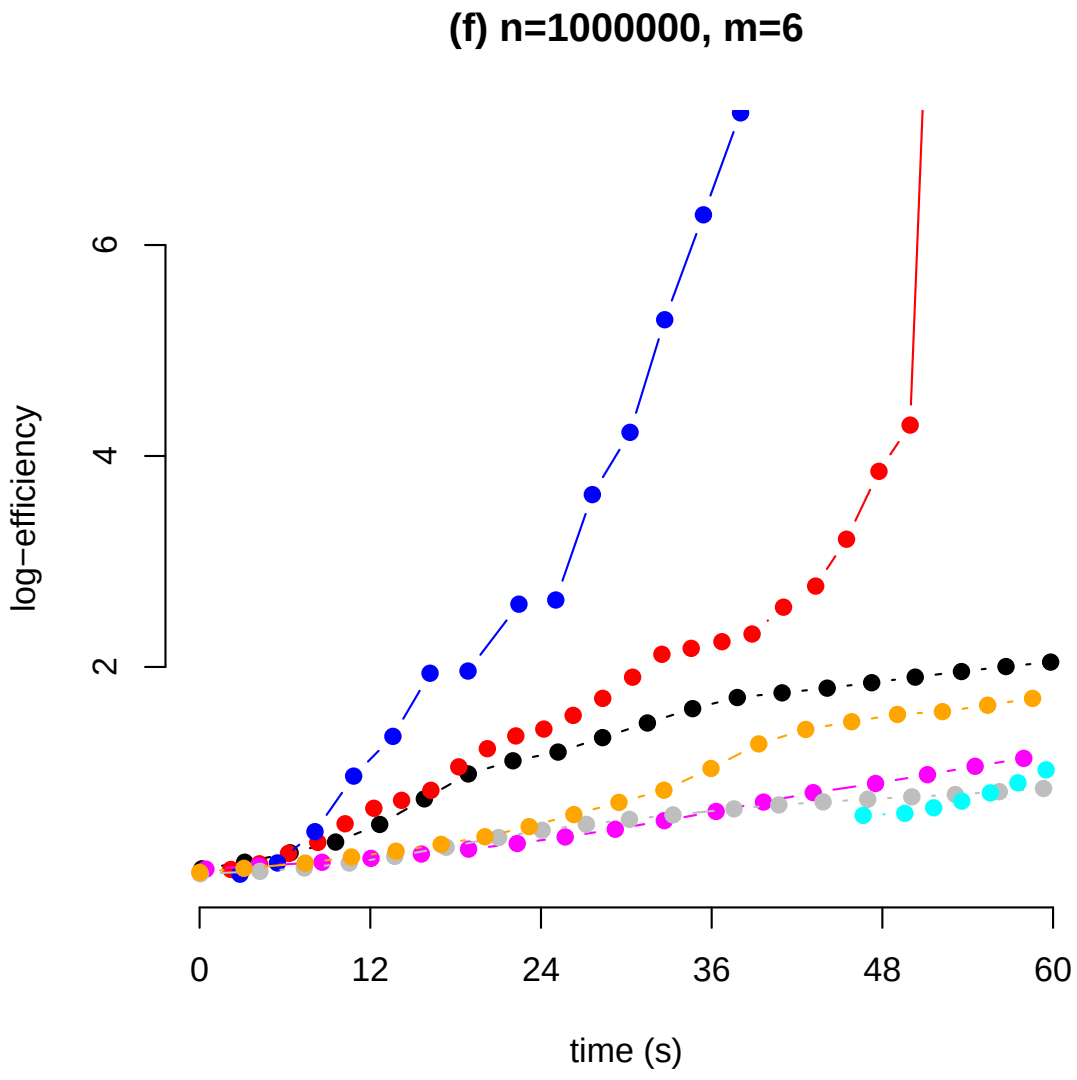

Figure: REX, SIM, COA, MUL, AWS, VEM, VDM

Final remarks

Approximate *D*-optimality on a finite design space is fundamental, not only because of its central position in optimal design of experiments, but also because it is equivalent to the problem of computing the minimum-volume enclosing ellipsoid of multidimensional data-points; the algorithms for *D*-optimality will continue to be used.

Among the available algorithms, REX seems to be generally the best and VDM is usually the worst, but each algorithm has some advantages. In particular, for a large m, the MUL algorithm can be more efficient than all other algorithms (including REX).

The development of algorithms for *D*-optimality on finite design spaces is (and will be) “work in progress”. For instance, can we efficiently incorporate the deletion rules (Harman & Pronzato 2007) into the algorithms? Can we perhaps utilize parallel computing? Is it possible to use methods of machine learning to “teach” some of the algorithms to perform very efficient heuristic choices (e.g., REX gives much freedom of choice that could be exploited).

Play with REX at

`https://optdesign.shinyapps.io/rexalg/`

or download the (short) R code of REX at

`http://www.iam.fmph.uniba.sk/ospm/Harman/design/`

Thank you for attention!