A genetic algorithm approach for the approximation of the joint spectral radius

Chia-Tche Chang and Vincent D. Blondel

Université catholique de Louvain, Belgium

30th Benelux Meeting on Systems and Control
Lommel, March 16th, 2011

Overview

Motivation: switching linear iterations

Discrete-time linear system of the form:

$$x(t+1) = Ax(t), \qquad A \in \mathbb{R}^{n\times n} \qquad\qquad \text{for all } t.$$

Growth and stability ruled by the spectral radius $\rho(A)$.

- ◇ No restriction on the sequence of matrices A_t.
- ◇ Switching depending on the state, external signal, due to asynchronism, randomness...

Motivation: switching linear iterations

Discrete-time linear system of the form:

$$x(t+1) = A_t x(t), \qquad A_t \in \Sigma \subset \mathbb{R}^{n\times n} \quad \text{for all } t.$$

Growth and stability ruled by the joint spectral radius $\rho(\Sigma)$.

- ◇ No restriction on the sequence of matrices A_t.
- ◇ Switching depending on the state, external signal, due to asynchronism, randomness...

Motivation: switching linear iterations

Discrete-time linear system of the form:

$$x(t+1) = A_t x(t), \qquad A_t \in \Sigma \subset \mathbb{R}^{n\times n} \quad \text{for all } t.$$

Growth and stability ruled by the joint spectral radius $\rho(\Sigma)$.

- ◇ No restriction on the sequence of matrices A_t.
- ◇ Switching depending on the state, external signal, due to asynchronism, randomness...

The joint spectral radius

- For a single matrix A: $\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k}$ (Gelfand).
- For a set Σ of matrices:
  - Joint spectral radius (Rota, Strang)

$$\widehat{\rho}(\Sigma) = \lim_{k\to\infty} \max\{\|A_{i_1}\dots A_{i_k}\|^{1/k} \mid A_i \in \Sigma\}.$$

  - Generalized spectral radius (Daubechies, Lagarias)

$$\rho(\Sigma) = \limsup_{k\to\infty} \max\{\rho(A_{i_1}\dots A_{i_k})^{1/k} \mid A_i \in \Sigma\}.$$

The joint spectral radius

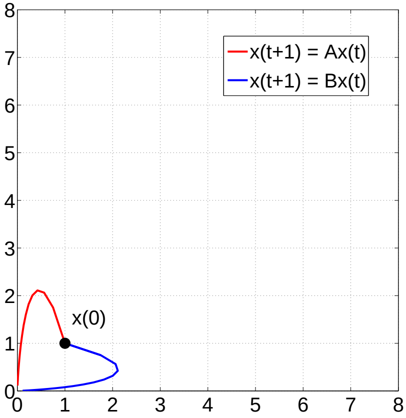
- ◇ For a single matrix A: $\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k}$ (Gelfand).
- ◇ For a set Σ of matrices:
  - Joint spectral radius (Rota, Strang)

    $$\widehat{\rho}(\Sigma) = \lim_{k\to\infty} \max\{\|A_{i_1} \dots A_{i_k}\|^{1/k} \mid A_i \in \Sigma\}.$$

  - Generalized spectral radius (Daubechies, Lagarias)

    $$\rho(\Sigma) = \limsup_{k\to\infty} \max\{\rho(A_{i_1} \dots A_{i_k})^{1/k} \mid A_i \in \Sigma\}.$$

The joint spectral radius



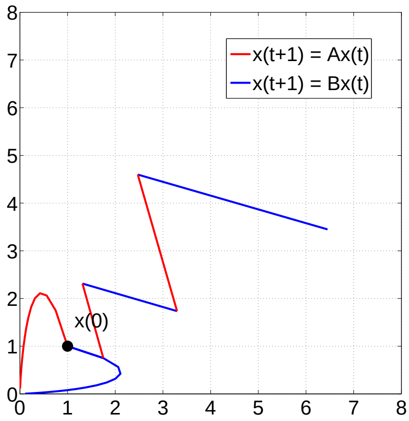
$$A = \begin{pmatrix} \frac{3}{4} & 0 \\ 1 & \frac{3}{4} \end{pmatrix}$$
$$B = \begin{pmatrix} \frac{3}{4} & 1 \\ 0 & \frac{3}{4} \end{pmatrix}$$

A, B are both stable:
$$\rho(A),\ \rho(B) < 1$$

$$\rho(AB) = \frac{17}{16} + \frac{1}{4}\sqrt{13} > 1$$

$\rho(\Sigma) \geqslant \rho(AB)^{\frac{1}{2}} > 1$ (unstable)

The joint spectral radius



$$A = \begin{pmatrix} \frac{3}{4} & 0 \\ 1 & \frac{3}{4} \end{pmatrix}$$

$$B = \begin{pmatrix} \frac{3}{4} & 1 \\ 0 & \frac{3}{4} \end{pmatrix}$$

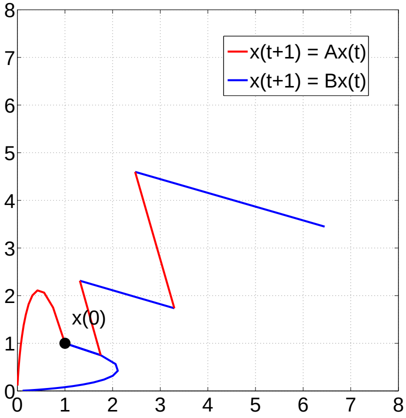
A, B are both stable:

$$\rho(A),\ \rho(B) < 1$$

$$\rho(AB) = \frac{17}{16} + \frac{1}{4}\sqrt{13} > 1$$

$\rho(\Sigma) \geqslant \rho(AB)^{\frac{1}{2}} > 1$ (unstable)

The joint spectral radius



$$A = \begin{pmatrix} \frac{3}{4} & 0 \\ 1 & \frac{3}{4} \end{pmatrix}$$
$$B = \begin{pmatrix} \frac{3}{4} & 1 \\ 0 & \frac{3}{4} \end{pmatrix}$$

A, B are both stable:

$$\rho(A),\ \rho(B) < 1$$

$$\rho(AB) = \frac{17}{16} + \frac{1}{4}\sqrt{13} > 1$$

$$\rho(\Sigma) = \rho(AB)^{\frac{1}{2}} > 1 \text{ (unstable)}$$

The joint spectral radius is difficult to evaluate

In the last example, $\rho(\Sigma) = \rho(AB)^{\frac{1}{2}}$.

Finiteness property: Maximal growth rate given by a periodic product.

Finiteness conjecture (false): All sets $\Sigma \subset \mathbb{R}^{n \times n}$ possess the FP.

$\diamond$ Approximating the JSR is NP-Hard, even for binary matrices.

$\diamond$ Determining if $\rho(\Sigma) \leqslant 1$ is undecidable, even for nonnegative rational matrices.

The joint spectral radius is difficult to evaluate

In the last example, $\rho(\Sigma) = \rho(AB)^{\frac{1}{2}}$.

Finiteness property: Maximal growth rate given by a periodic product.

Finiteness conjecture (false): All sets $\Sigma \subset \mathbb{R}^{n \times n}$ possess the FP.

$\diamond$ Approximating the JSR is NP-Hard, even for binary matrices.

$\diamond$ Determining if $\rho(\Sigma) \leqslant 1$ is undecidable, even for nonnegative rational matrices.

How to compute the joint spectral radius?

Define:

$$\widehat{\rho}_k(\Sigma) = \max\{\|A_{i_1} \dots A_{i_k}\|^{1/k} \mid A_i \in \Sigma\},$$
$$\rho_k(\Sigma) = \max\{\rho(A_{i_1} \dots A_{i_k})^{1/k} \mid A_i \in \Sigma\}.$$

Recall that by definition of the JSR:

$$\limsup_{k \to \infty} \rho_k(\Sigma) = \rho(\Sigma) = \lim_{k \to \infty} \widehat{\rho}_k(\Sigma).$$

We have:

$$\rho_k(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_k(\Sigma).$$

Values of $\widehat{\rho}_k(\Sigma)$ depends on the norm!

How to compute the joint spectral radius?

Define:

$$\widehat{\rho}_k(\Sigma) = \max\{\|A_{i_1} \dots A_{i_k}\|^{1/k} \mid A_i \in \Sigma\},$$

$$\rho_k(\Sigma) = \max\{\rho(A_{i_1} \dots A_{i_k})^{1/k} \mid A_i \in \Sigma\}.$$

Recall that by definition of the JSR:

$$\limsup_{k\to\infty} \rho_k(\Sigma) = \rho(\Sigma) = \lim_{k\to\infty} \widehat{\rho}_k(\Sigma).$$

We have:

$$\rho_k(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_k(\Sigma).$$

Values of $\widehat{\rho}_k(\Sigma)$ depends on the norm!

First approach: consider a large set of products

For all k, we have the converging bounds $\rho_k(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_k(\Sigma)$.

Brute-force is only reasonable for small problems but branch-and-bound approach is possible.

Gripenberg's algorithm: given ε, uses a branch-and-bound technique to return lower and upper bounds $\rho^- \leqslant \rho(\Sigma) \leqslant \rho^+$ with $\rho^+ - \rho^- \leqslant \varepsilon$.

- ◇ Guaranteed converging bounds at each step.
- ◇ Convergence may be slow depending on the norm used.
- ◇ Number of steps to reach an interval of length ε is unknown.
- ◇ May require very long products.

First approach: consider a large set of products

For all k, we have the converging bounds $\rho_k(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_k(\Sigma)$.

Brute-force is only reasonable for small problems but branch-and-bound approach is possible.

Gripenberg's algorithm: given ε, uses a branch-and-bound technique to return lower and upper bounds $\rho^- \leqslant \rho(\Sigma) \leqslant \rho^+$ with $\rho^+ - \rho^- \leqslant \varepsilon$.

- Guaranteed converging bounds at each step.
- Convergence may be slow depending on the norm used.
- Number of steps to reach an interval of length ε is unknown.
- May require very long products.

First approach: consider a large set of products

For all k, we have the converging bounds $\rho_k(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_k(\Sigma)$.

Brute-force is only reasonable for small problems but branch-and-bound approach is possible.

Gripenberg's algorithm: given ε, uses a branch-and-bound technique to return lower and upper bounds $\rho^- \leqslant \rho(\Sigma) \leqslant \rho^+$ with $\rho^+ - \rho^- \leqslant \varepsilon$.

- ◇ Guaranteed converging bounds at each step.
- ◇ Convergence may be slow depending on the norm used.
- ◇ Number of steps to reach an interval of length ε is unknown.
- ◇ May require very long products.

Second approach: consider a large set of norms

Norm dependency of the upper bounds $\widehat{\rho}_k(\Sigma)$
$\rightarrow$ try to find a norm giving good bounds with short products.

A norm is extremal if $\rho(\Sigma) = \max_{A_i \in \Sigma} \|A_i\|$ (product of length 1).

It can be proven that $\rho(\Sigma) = \inf_{\|\cdot\|} \max_{A_i \in \Sigma} \|A_i\|$.

Idea: minimize over a well-chosen set of norms.

Second approach: consider a large set of norms

Norm dependency of the upper bounds $\widehat{\rho}_k(\Sigma)$
$\rightarrow$ try to find a norm giving good bounds with short products.

A norm is extremal if $\rho(\Sigma) = \max_{A_i \in \Sigma} \|A_i\|$ (product of length 1).

It can be proven that $\rho(\Sigma) = \inf_{\|\cdot\|} \max_{A_i \in \Sigma} \|A_i\|$.

Idea: minimize over a well-chosen set of norms.

Finding an ellipsoidal norm using optimization

Ellipsoidal vector norm: $\|x\|_P = \sqrt{x^T P x}$ for a given $P \succ 0$.

Ellipsoidal norm approximation: $\widehat{\rho}_{ell}(\Sigma) = \inf_{P \succ 0} \max_{A_i \in \Sigma} \|A_i\|_P$.

Upper bound $\widehat{\rho}_{ell}(\Sigma)$ can be computed using semidefinite optimization:

$$\widehat{\rho}_{ell}(\Sigma) = \min_{\gamma \in \mathbb{R}, P \succ 0} \left\{ \gamma \mid \gamma^2 P - A_i^T P A_i \succeq 0 \text{ for all } A_i \in \Sigma \right\}.$$

- Guarantee: $\frac{1}{\sqrt{\max\{n, |\Sigma|\}}} \widehat{\rho}_{ell}(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_{ell}(\Sigma)$.
- Extensions: polynomials and sum-of-squares, conic programming.
- May require solving a large SDP problem.
- Subject to numerical issues.

Finding an ellipsoidal norm using optimization

Ellipsoidal vector norm: $\|x\|_P = \sqrt{x^T P x}$ for a given $P \succ 0$.

Ellipsoidal norm approximation: $\widehat{\rho}_{ell}(\Sigma) = \inf_{P \succ 0} \max_{A_i \in \Sigma} \|A_i\|_P$.

Upper bound $\widehat{\rho}_{ell}(\Sigma)$ can be computed using semidefinite optimization:

$$\widehat{\rho}_{ell}(\Sigma) = \min_{\gamma \in \mathbb{R}, P \succ 0} \left\{ \gamma \mid \gamma^2 P - A_i^T P A_i \succeq 0 \text{ for all } A_i \in \Sigma \right\}.$$

- Guarantee: $\frac{1}{\sqrt{\max\{n, |\Sigma|\}}} \widehat{\rho}_{ell}(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_{ell}(\Sigma)$.
- Extensions: polynomials and sum-of-squares, conic programming.
- May require solving a large SDP problem.
- Subject to numerical issues.

Finding an ellipsoidal norm using optimization

Ellipsoidal vector norm: $\|x\|_P = \sqrt{x^T P x}$ for a given $P \succ 0$.

Ellipsoidal norm approximation: $\widehat{\rho}_{ell}(\Sigma) = \inf_{P \succ 0} \max_{A_i \in \Sigma} \|A_i\|_P$.

Upper bound $\widehat{\rho}_{ell}(\Sigma)$ can be computed using semidefinite optimization:

$$\widehat{\rho}_{ell}(\Sigma) = \min_{\gamma \in \mathbb{R}, P \succ 0} \left\{ \gamma \mid \gamma^2 P - A_i^T P A_i \succeq 0 \text{ for all } A_i \in \Sigma \right\}.$$

- ◇ Guarantee: $\frac{1}{\sqrt{\max\{n, |\Sigma|\}}} \widehat{\rho}_{ell}(\Sigma) \leqslant \rho(\Sigma) \leqslant \widehat{\rho}_{ell}(\Sigma)$.
- ◇ Extensions: polynomials and sum-of-squares, conic programming.
- ◇ May require solving a large SDP problem.
- ◇ Subject to numerical issues.

Third approach: build an extremal norm

Instead of considering a large set of norms and “hope” that it contains an extremal one, try to directly construct such an extremal norm.

Several algorithms, e.g., Kozyakin’s LR and MR-procedures use this approach.

- ◇ Guaranteed converging bounds at each iteration in theory.
- ◇ Most algorithms require manipulation of geometric objects (polytopes, unit balls of norms, . . .)
- ◇ Practical convergence may be slow due to discretization and numerical problems.

Third approach: build an extremal norm

Instead of considering a large set of norms and “hope” that it contains an extremal one, try to directly construct such an extremal norm.

Several algorithms, e.g., Kozyakin’s LR and MR-procedures use this approach.

- ◇ Guaranteed converging bounds at each iteration in theory.
- ◇ Most algorithms require manipulation of geometric objects (polytopes, unit balls of norms, . . .)
- ◇ Practical convergence may be slow due to discretization and numerical problems.

Why using a genetic algorithm?

Most “classical” methods have some theoretical guarantees but are often too slow and/or fail due to numerical problems if we want a good accuracy.

Here, we are willing to drop guarantees* in exchange of a fast running algorithm able to handle reasonably large size problems.

(*) Only return a lower bound on the JSR but with no a priori guarantee on its quality.

Why using a genetic algorithm?

Most “classical” methods have some theoretical guarantees but are often too slow and/or fail due to numerical problems if we want a good accuracy.

Here, we are willing to drop guarantees* in exchange of a fast running algorithm able to handle reasonably large size problems.

(*) Only return a lower bound on the JSR but with no a priori guarantee on its quality.

What is a genetic algorithm?

GA is a stochastic beam-search evolutionary method...

- ◇ Stochastic: include random elements.
- ◇ Beam-search: keep a set of candidates at each iteration.
- ◇ Evolutionary: generate new candidates by combining current ones.

Many variants are possible for the generation of new candidates from old ones.

Application to the joint spectral radius

◇ Preprocess and generate an initial population of size M.

- Evaluate all products of length $\leqslant k$ for some k.
- Best product gives an initial lower bound on the JSR.
- Generate M random products of length $\leqslant K = 2k$ as initial population.

At each generation:

◇ Evaluate the performance of all population members.

◇ Generate the new population based on the current one.

◇ Apply random mutations with some probability.

◇ Enlarge the search space if no improvement is done.

Application to the joint spectral radius

- ◇ Preprocess and generate an initial population of size M.

  At each generation:

- ◇ Evaluate the performance of all population members.
  - If the bound on the JSR is improved, explore the neighborhood of the corresponding product (Levenshtein distance of 1).
  - If a better product is found in this neighborhood, insert it in the population, replacing the worst one.
- ◇ Generate the new population based on the current one.
- ◇ Apply random mutations with some probability.
- ◇ Enlarge the search space if no improvement is done.

Application to the joint spectral radius

- ◇ Preprocess and generate an initial population of size M.

  At each generation:
- ◇ Evaluate the performance of all population members.
- ◇ Generate the new population based on the current one.
  - Best products are kept (elitism).
  - New products are produced by swapping good ones:
    $$A_1A_2A_3A_4A_5 \oplus B_1B_2B_3B_4B_5 \longrightarrow A_1A_2B_3B_4B_5.$$
  - Others are produced by mixing old products:
    $$A_1A_2A_3A_4A_5 \otimes B_1B_2B_3B_4B_5 \longrightarrow A_1B_2B_3A_4B_5.$$
  - New random products are inserted to ensure exploration.
- ◇ Apply random mutations with some probability.
- ◇ Enlarge the search space if no improvement is done.

Application to the joint spectral radius

- ◇ Preprocess and generate an initial population of size M.

  At each generation:
- ◇ Evaluate the performance of all population members.
- ◇ Generate the new population based on the current one.
- ◇ Apply random mutations with some probability.
  - ▸ Randomly modify some parts of a small number of products to ensure exploration.
- ◇ Enlarge the search space if no improvement is done.

Application to the joint spectral radius

- Preprocess and generate an initial population of size M.

At each generation:

- Evaluate the performance of all population members.
- Generate the new population based on the current one.
- Apply random mutations with some probability.
- Enlarge the search space if no improvement is done.
    - If the bound keeps stalling for T_1 generations, increase the maximum product length K and try again.
    - If there is still no improvement for T_2 generations, abort the algorithm and return the best bound found.

A first numerical example

Test sets: 100 sets of randomly-generated matrix with entries in $[-5, 5]$

Smaller problems: $|\Sigma| = 2$, $\Sigma \subset \mathbb{R}^{2\times 2}$.

Comparison of lower bounds given by brute-force approach, Gripenberg's algorithm (1st approach), LR/MR-procedures (3rd approach), and genetic algorithm.

Performance mesured by the number of times the algorithm returns the best bound among all algorithms, within a given tolerance.

A first numerical example

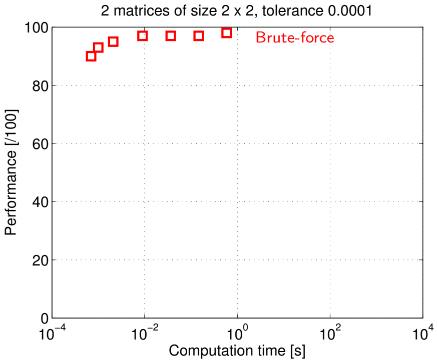
Test sets: 100 sets of randomly-generated matrix with entries in $[-5, 5]$

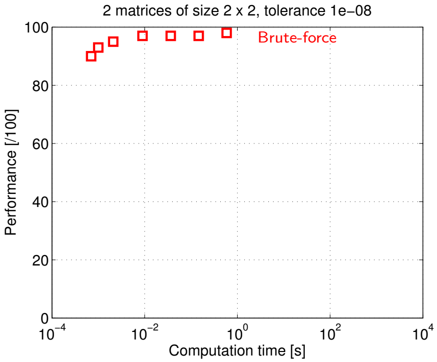
Smaller problems: $|\Sigma| = 2$, $\Sigma \subset \mathbb{R}^{2\times 2}$.

Comparison of lower bounds given by brute-force approach, Gripenberg's algorithm (1st approach), LR/MR-procedures (3rd approach), and genetic algorithm.

Performance mesured by the number of times the algorithm returns the best bound among all algorithms, within a given tolerance.

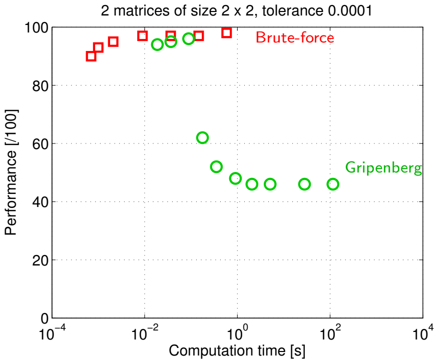
Small sets (2 random matrices of size 2x2)

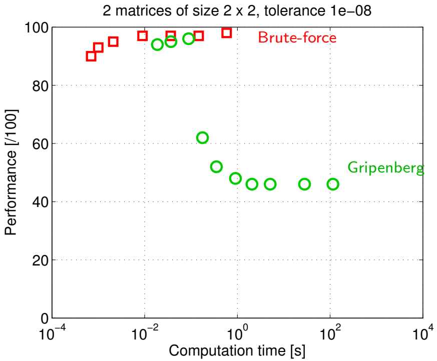




Brute-force: products of length $2 \sim 12$, manageable due to small size

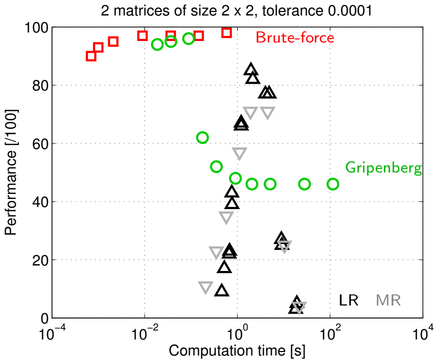
Small sets (2 random matrices of size 2x2)

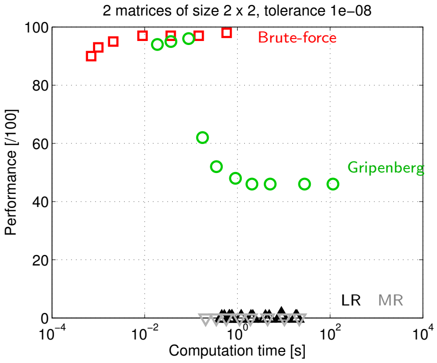




Gripenberg: $100 \sim 10^5$ evaluations, fails due to numerical accuracy

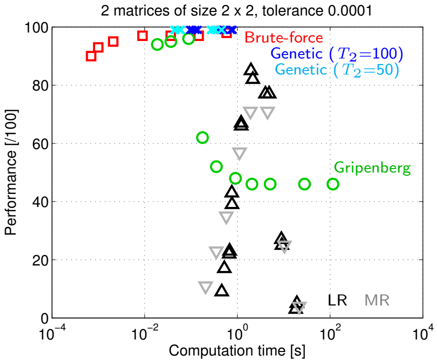
Small sets (2 random matrices of size 2x2)

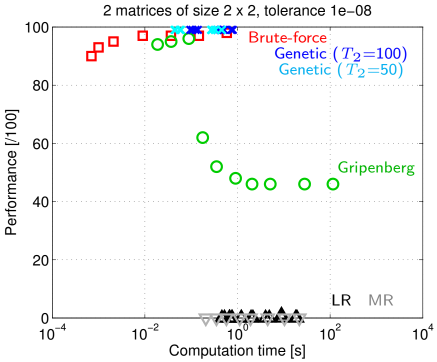




LR/MR-procedures: $500 \sim 10^5$ points, imprecise and numerical issues

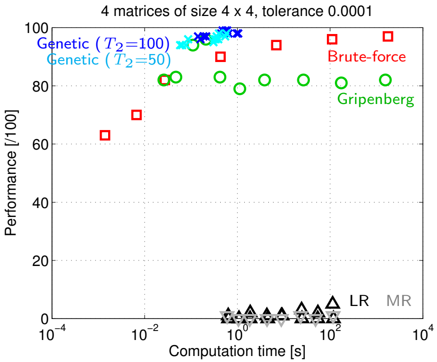
Small sets (2 random matrices of size 2x2)

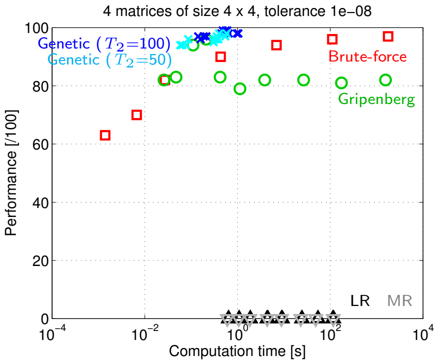




Genetic: population size $15 \sim 100$, stalling threshold $T_1 \in \{10, 15\}$.

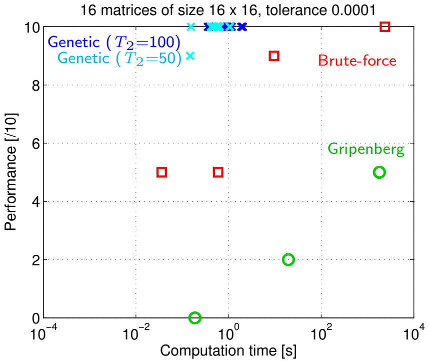
Larger sets (4 random matrices of size 4x4)

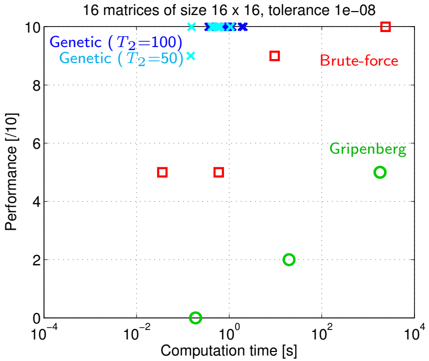




Running time of the genetic algorithm is similar to the smaller problem.

Even larger sets (16 capacity matrices of size 16x16)





Genetic algorithm can manage the problem size increase.
LR/MR-procedures require too much memory.

Conclusions

◇ The approximation of the JSR is a difficult computational problem.

◇ “Classical” methods have theoretical guarantees but are unable to handle large size problems in practice (computation time, memory usage, numerical issues).

◇ The genetic algorithm has no a priori guarantee but performs very well with a low running time.

Further work: parameter selection, other joint spectral quantities, . . .

Conclusions

- ◇ The approximation of the JSR is a difficult computational problem.
- ◇ "Classical" methods have theoretical guarantees but are unable to handle large size problems in practice (computation time, memory usage, numerical issues).
- ◇ The genetic algorithm has no a priori guarantee but performs very well with a low running time.

Further work: parameter selection, other joint spectral quantities, ...