Compress Then Test: Powerful Kernel Testing in Near-linear Time

Carles Domingo-Enrich

NYU

Raaz Dwivedi

HARVARD UNIVERSITY

Massachusetts Institute of Technology

CORNELL TECH

Lester Mackey

Microsoft Research

JSM 2023

Two-sample testing

Test null hypothesis: $\mathscr{H}_0 : \mathbb{P} = \mathbb{Q}$

with $\mathbf{X}_n = (X_i)_{1 \leq i \leq n}$ i.i.d. sample of $\mathbb{P}$

$\mathbf{Y}_n = (Y_i)_{1 \leq i \leq n}$ i.i.d. sample of $\mathbb{Q}$

Parametric vs non-parametric test

Parametric vs non-parametric test

- An alternate representation of $\mathscr{H}_0 : \mathbb{P} = \mathbb{Q}$

$$\mathbb{E}_{X \sim \mathbb{P}} f(X) = \mathbb{E}_{Y \sim \mathbb{Q}} f(Y) \quad \text{for all } f \in \mathscr{F} \text{ (a class of test functions)}$$

Parametric vs non-parametric test

- An alternate representation of $\mathscr{H}_0 : \mathbb{P} = \mathbb{Q}$

$$\mathbb{E}_{X\sim\mathbb{P}} f(X) = \mathbb{E}_{Y\sim\mathbb{Q}} f(Y) \quad \text{for all } f \in \mathscr{F} \text{ (a class of test functions)}$$

- Parametric test: When $\mathscr{F}$ is finite-dimensional
  (e.g., $\mathscr{F} = \{X, X^2\}$ for testing Gaussian $\mathbb{P}, \mathbb{Q}$)

Parametric vs non-parametric test

- An alternate representation of $\mathscr{H}_0 : \mathbb{P} = \mathbb{Q}$

$$\mathbb{E}_{X\sim\mathbb{P}} f(X) = \mathbb{E}_{Y\sim\mathbb{Q}} f(Y) \quad \text{for all } f \in \mathscr{F} \text{ (a class of test functions)}$$

- Parametric test: When $\mathscr{F}$ is finite-dimensional
  (e.g., $\mathscr{F} = \{X, X^2\}$ for testing Gaussian $\mathbb{P}, \mathbb{Q}$)

- Non-parametric test: When $\mathscr{F}$ is infinite-dimensional
  (e.g., $\mathscr{F}$ = unit ball of a reproducing kernel Hilbert space)

Reproducing kernel Hilbert space

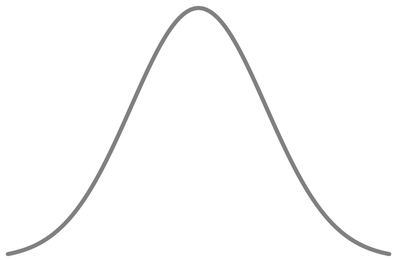
Reproducing kernel Hilbert space

- A symmetric function $\mathbf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, i.e., $\mathbf{k}(x, y) = \mathbf{k}(y, x)$ which is positive semidefinite, is called a reproducing kernel

Reproducing kernel Hilbert space

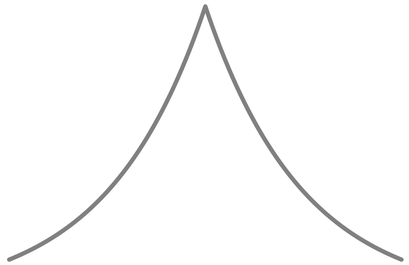
- A symmetric function $\mathbf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, i.e., $\mathbf{k}(x, y) = \mathbf{k}(y, x)$ which is positive semidefinite, is called a reproducing kernel

Gaussian



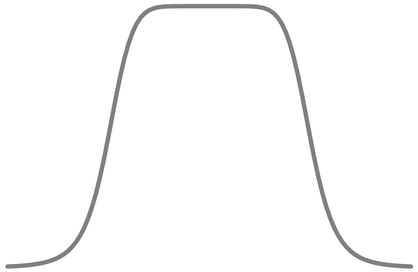
$\mathbf{k}(x, y) = e^{-\|x-y\|^2}$

Matérn



$\mathbf{k}(x, y) = e^{-\|x-y\|}$

Inverse multiquadric



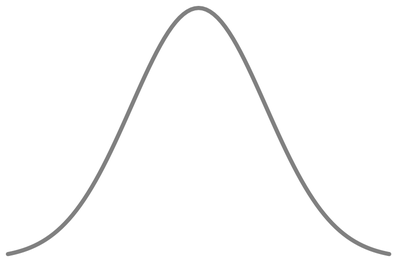
$\mathbf{k}(x, y) = (1 + \|x - y\|)^{-4}$

Reproducing kernel Hilbert space

- A symmetric function $\mathbf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, i.e., $\mathbf{k}(x, y) = \mathbf{k}(y, x)$ which is positive semidefinite, is called a reproducing kernel

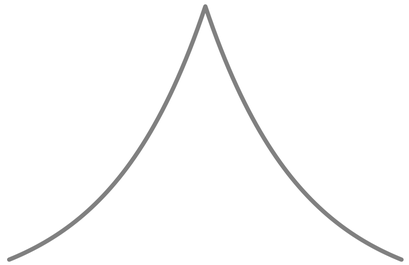
- Reproducing kernel Hilbert space = Closure of linear span of $\{\mathbf{k}(x, \cdot\,)\}$

Gaussian



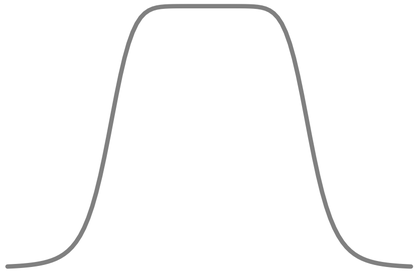
$\mathbf{k}(x, y) = e^{-\|x-y\|^2}$

Matérn



$\mathbf{k}(x, y) = e^{-\|x-y\|}$

Inverse multiquadric



$\mathbf{k}(x, y) = (1 + \|x - y\|)^{-4}$

Kernel two-sample testing

- Reparametrized via maximum mean discrepancy (MMD) of a kernel $\mathbf{k}$

$$\mathscr{H}_0 : \mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \triangleq \sup_{\|f\|_{\mathbf{k}} \leq 1} \mathbb{E}_{X \sim \mathbb{P}} f(X) - \mathbb{E}_{Y \sim \mathbb{Q}} f(Y) = 0$$

Kernel two-sample test

Kernel two-sample test

- **MMD expression:**

$$\mathscr{H}_0 : \mathrm{MMD}^2_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \triangleq \mathbb{E}_{X,X' \sim \mathbb{P}} \mathbf{k}(X, X') + \mathbb{E}_{Y,Y' \sim \mathbb{P}} \mathbf{k}(Y, Y') - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}} \mathbf{k}(X, Y) = 0$$

Kernel two-sample test

- **MMD expression:**

$$\mathscr{H}_0 : \mathrm{MMD}^2_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \triangleq \mathbb{E}_{X,X' \sim \mathbb{P}} \mathbf{k}(X, X') + \mathbb{E}_{Y,Y' \sim \mathbb{P}} \mathbf{k}(Y, Y') - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}} \mathbf{k}(X, Y) = 0$$

- **Test statistic:**

$$\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) = \frac{\sum_{i,i'=1}^n \mathbf{k}(X_i, X_{i'})}{n^2} + \frac{\sum_{j,j'=1}^n \mathbf{k}(Y_j, Y_{j'})}{n^2} - \frac{2\sum_{i=1}^n \sum_{j=1}^n \mathbf{k}(X_i, Y_j)}{n^2}$$

Kernel two-sample test

- **MMD expression:**

$$\mathscr{H}_0 : \mathrm{MMD}^2_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \triangleq \mathbb{E}_{X,X' \sim \mathbb{P}} \mathbf{k}(X, X') + \mathbb{E}_{Y,Y' \sim \mathbb{P}} \mathbf{k}(Y, Y') - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}} \mathbf{k}(X, Y) = 0$$

- **Test statistic:**

$$\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) = \frac{\sum_{i,i'=1}^n \mathbf{k}(X_i, X_{i'})}{n^2} + \frac{\sum_{j,j'=1}^n \mathbf{k}(Y_j, Y_{j'})}{n^2} - \frac{2\sum_{i=1}^n \sum_{j=1}^n \mathbf{k}(X_i, Y_j)}{n^2}$$

- **Test:** If $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) < t_\alpha$ accept the null; else reject

Kernel two-sample test

- **MMD expression:**

$$\mathscr{H}_0 : \mathrm{MMD}^2_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \triangleq \mathbb{E}_{X,X' \sim \mathbb{P}} \mathbf{k}(X, X') + \mathbb{E}_{Y,Y' \sim \mathbb{P}} \mathbf{k}(Y, Y') - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}} \mathbf{k}(X, Y) = 0$$

- **Test statistic:**

$$\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) = \frac{\sum_{i,i'=1}^n \mathbf{k}(X_i, X_{i'})}{n^2} + \frac{\sum_{j,j'=1}^n \mathbf{k}(Y_j, Y_{j'})}{n^2} - \frac{2\sum_{i=1}^n \sum_{j=1}^n \mathbf{k}(X_i, Y_j)}{n^2}$$

n^2 **runtime**

- **Test:** If $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) < t_\alpha$ accept the null; else reject

Type 1 error and validity of test

- **Valid test:** Threshold t_α chosen such that

$$\text{Type 1 error} = \mathrm{Prob}_{\mathscr{H}_0}\left[\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \geq t_\alpha\right] \leq \alpha$$

$$\mathscr{H}_0 : \mathbb{P} = \mathbb{Q}$$

Two strategies to choose threshold t_α

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

  ✓ n^2 runtime

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

  ✓ n^2 runtime

  ◉ inexact with finite samples; asymptotics might not be Gaussian

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

  ✓ n^2 runtime

  ◉ inexact with finite samples; asymptotics might not be Gaussian

- **Permutation tests:** Use permutations to simulate the distribution of test statistics under the null

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

  ✓ n^2 runtime

  ◉ inexact with finite samples; asymptotics might not be Gaussian

- **Permutation tests:** Use permutations to simulate the distribution of test statistics under the null

  ✓ exact type 1 error control in finite samples

Two strategies to choose threshold t_α

- **Asymptotic tests:** Use quantiles from the asymptotic distribution of test statistics under the null

  ✓ n^2 runtime

  ◉ inexact with finite samples; asymptotics might not be Gaussian

- **Permutation tests:** Use permutations to simulate the distribution of test statistics under the null

  ✓ exact type 1 error control in finite samples

  ◉ $n^2\mathscr{L}$ runtime, when doing $\mathscr{L}$ permutations (typically $\mathscr{L} \sim$ 100s)

Power of the test and MMD separation

Power of the test and MMD separation

- Power of the test = 1 − Type 2 error

$\mathscr{H}_1 : \mathbb{P} \neq \mathbb{Q}$

$$= \mathrm{Prob}_{\mathscr{H}_1}\left[\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \geq t_\alpha\right]$$

Power of the test and MMD separation

- Power of the test = 1 − Type 2 error $\qquad \mathscr{H}_1 : \mathbb{P} \neq \mathbb{Q}$

$$= \mathrm{Prob}_{\mathscr{H}_1}\left[\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \geq t_\alpha\right]$$

- MMD separation rate$_n$

  = Minimum value of $\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q})$ such that power of the test is bounded away from 0

Test name	MMD separation	Runtime
Complete MMD (Gretton et al., 2012a)	$n^{-\frac{1}{2}}$	n^2

Our goal: Speed up the test runtime while maintaining

1. **validity for all sample sizes** (exact type 1 error)

2. **small MMD separation rate** (high power, low type 2 error)

Test name	MMD separation	Runtime
Complete MMD (Gretton et al., 2012a)	$n^{-\frac{1}{2}}$	n^2
Block MMD (Zaremba et al., 2013)	$\frac{1}{(n/B)^{1/4}\sqrt{B}} = (Bn)^{-1/4}$	$\frac{n}{B}B^2 = Bn$

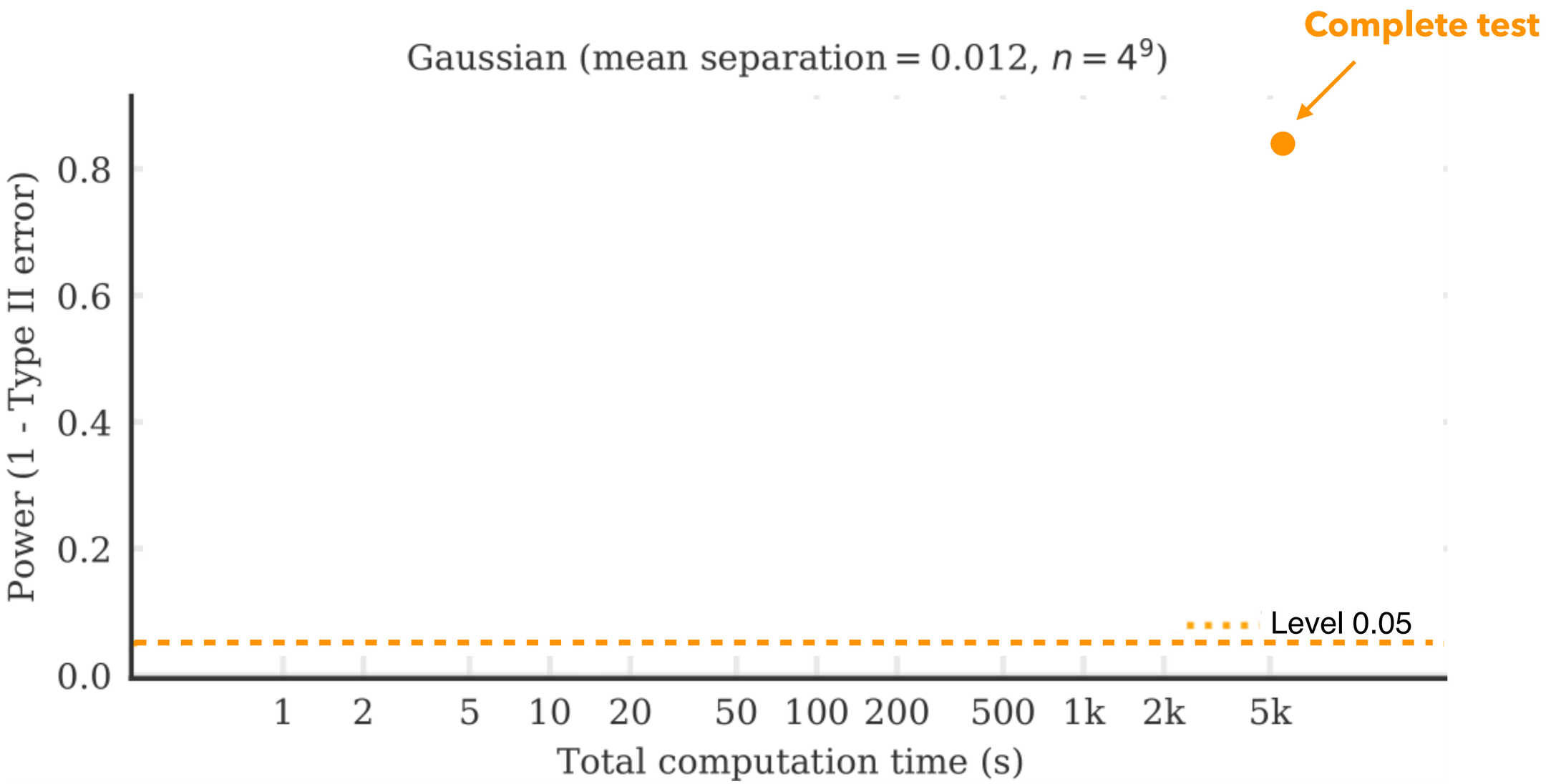
B = Number of blocks used in Block MMD test

Test name	MMD separation	Runtime
Complete MMD (Gretton et al., 2012a)	$n^{-\frac{1}{2}}$	n^2
Block MMD (Zaremba et al., 2013)	$(Bn)^{-\frac{1}{4}}$	Bn
Incomplete MMD (Yamada et al., 2019)	$\ell^{-\frac{1}{4}}$	ℓ

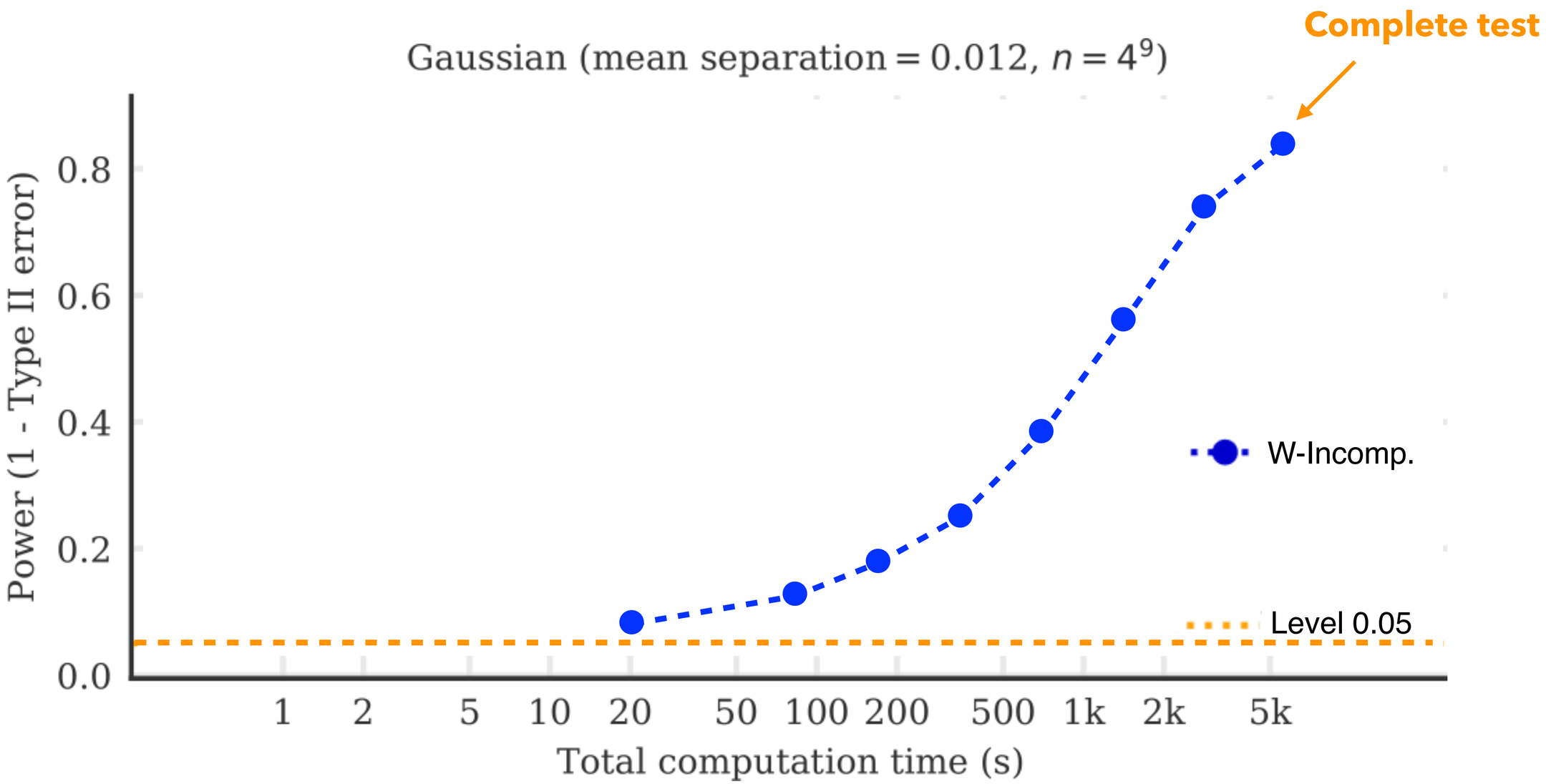
B = Number of blocks used in Block MMD test

ℓ = Number of ordered index pairs in Incomplete MMD test

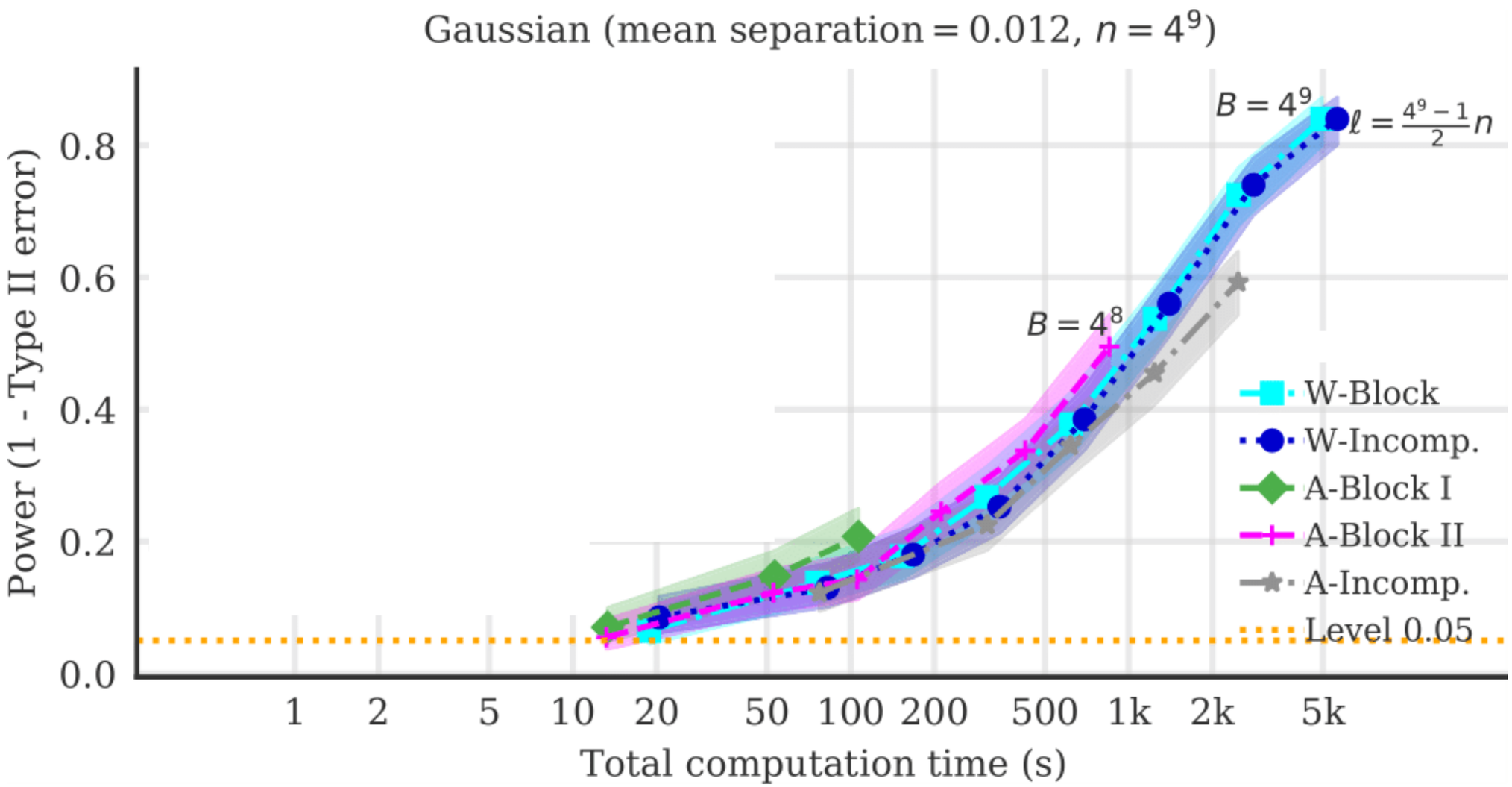
Empirical power-runtime tradeoff curves



Empirical power-runtime tradeoff curves

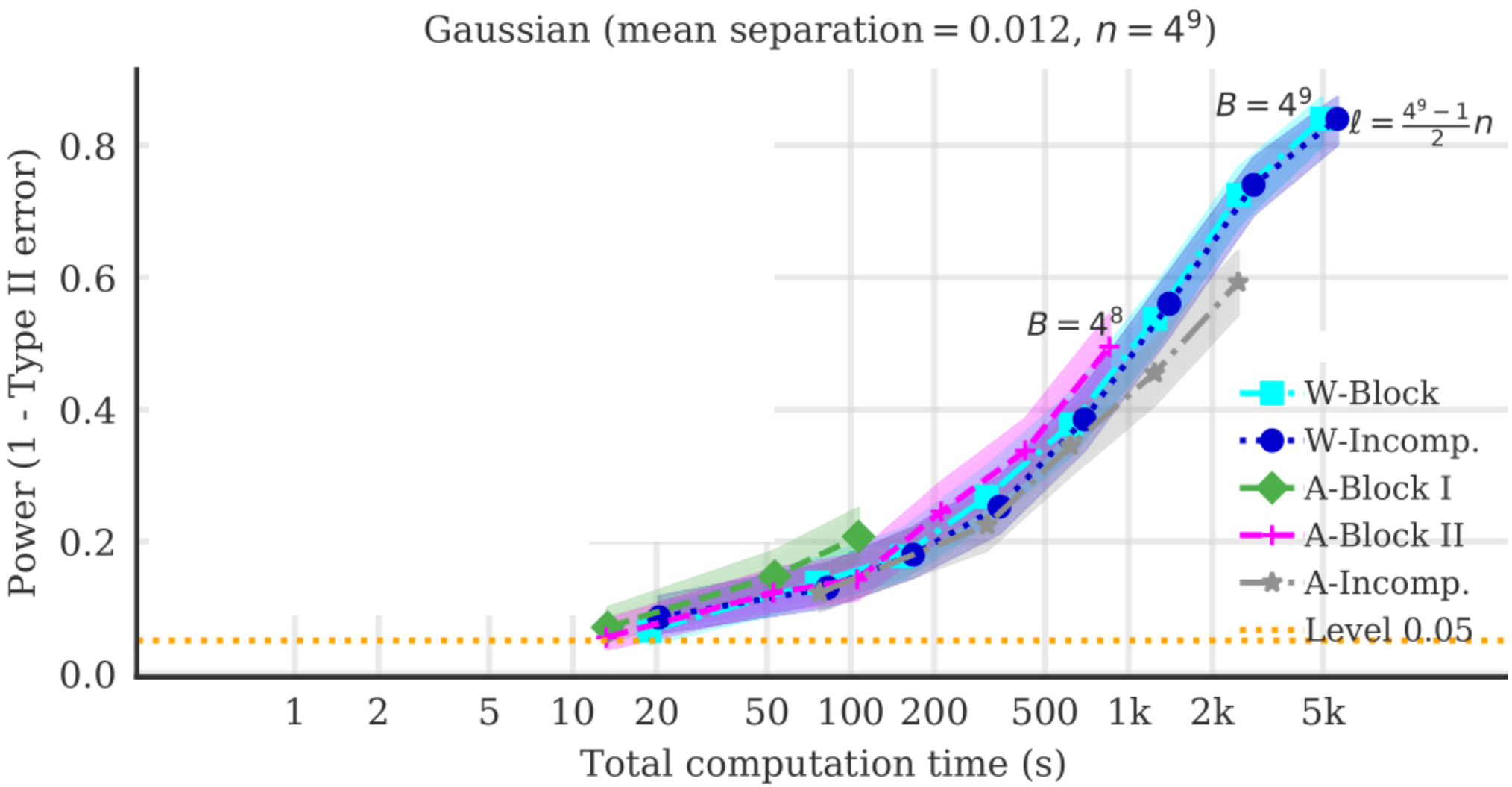


Prior strategies to speed up testing

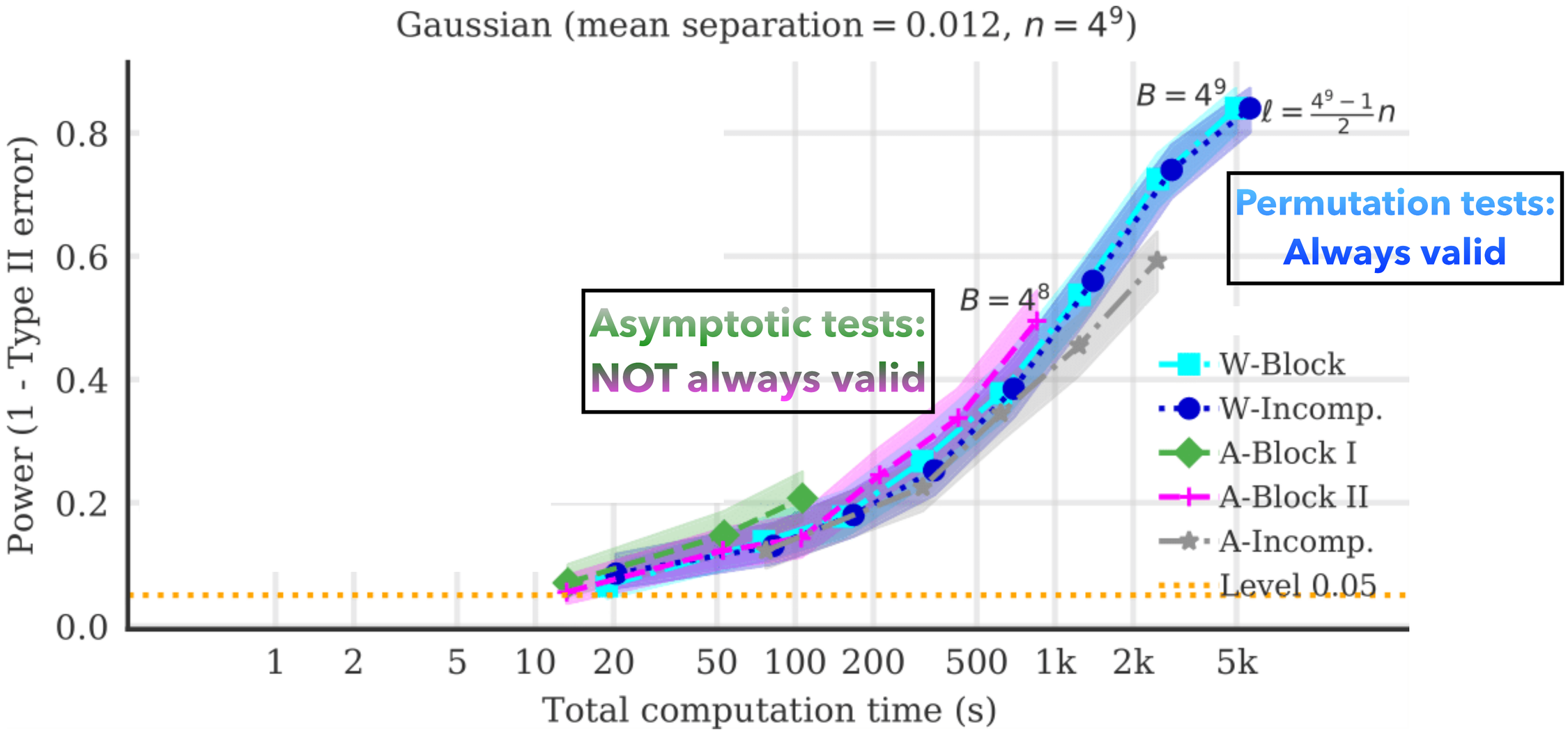


Prior strategies to speed up testing:

Faster runtimes, but only at the loss of power

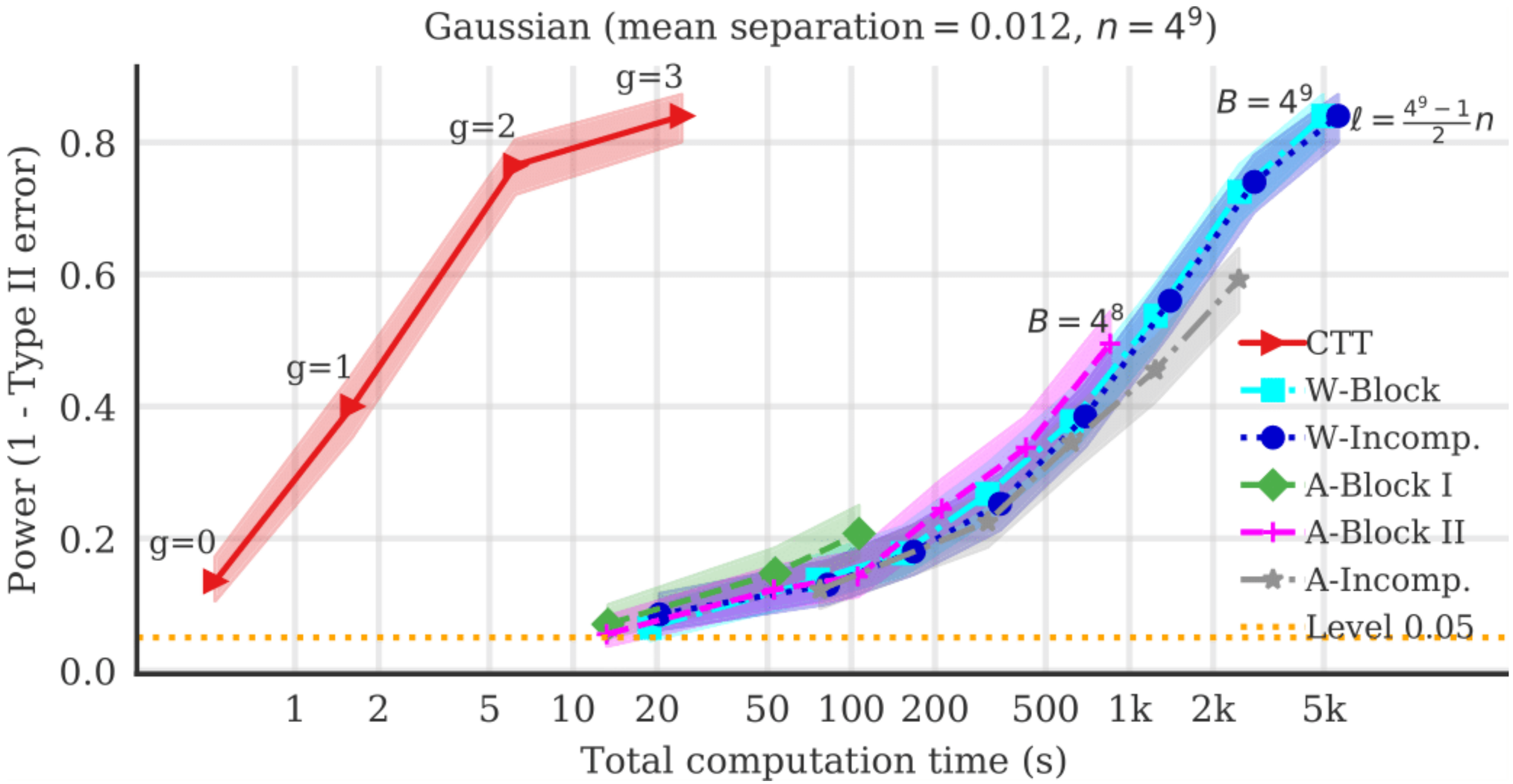


Prior strategies to speed up testing:
Faster runtimes, but only at the loss of power



Our solution: Compress Then Test

Best of <u>three</u> worlds: Fast + Always valid + High power



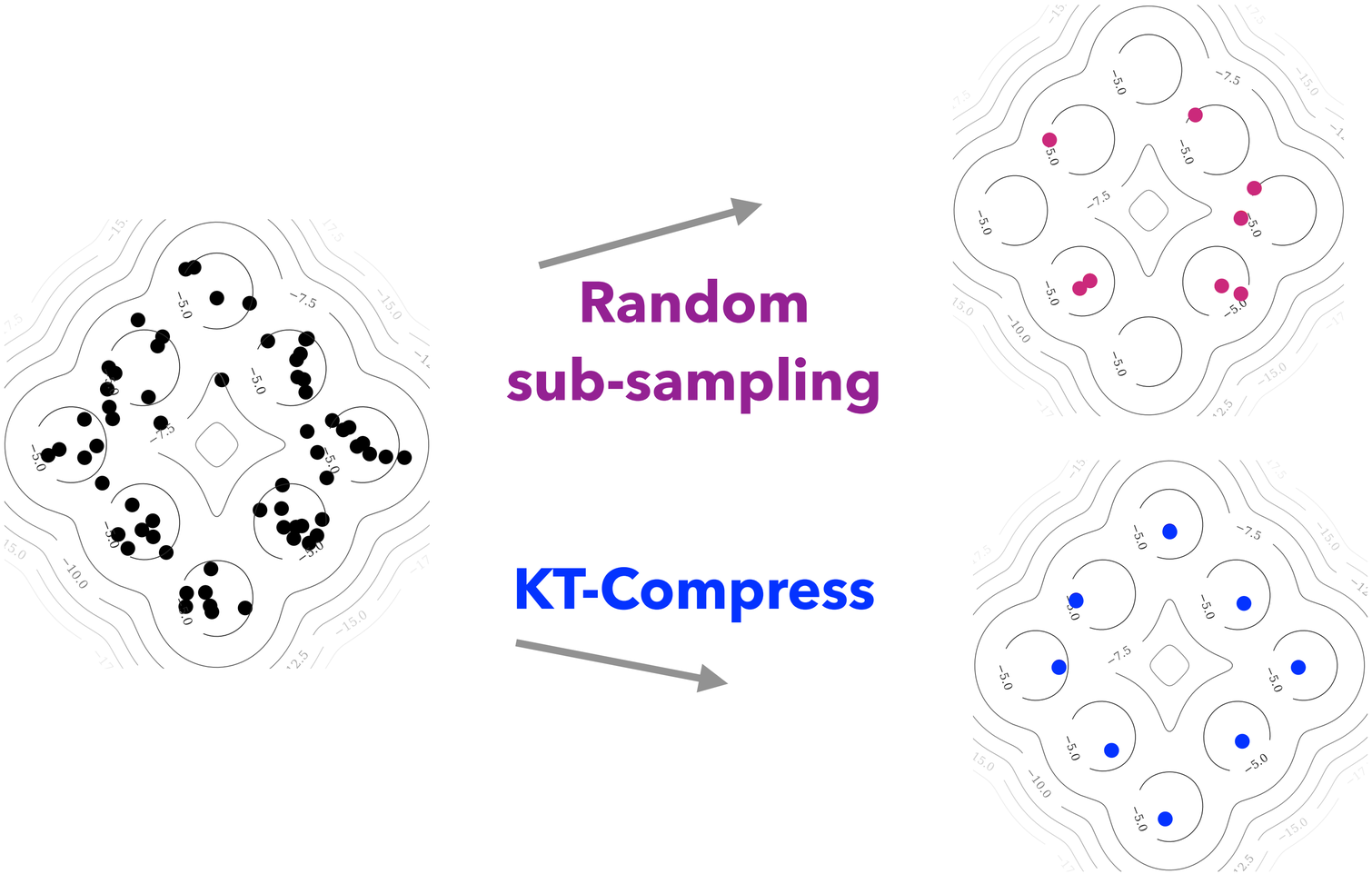
Key idea: Compress the data and then test

Compress $\mathbf{X}_n$ to $\hat{\mathbf{X}}_m$ of size $m = \sqrt{n}$ with small $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m)$

	Random subsampling	KT-Compress (Shetty-**Dwivedi**-Mackey, 2022)
$\mathrm{MMD}_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m)$	$O(n^{-1/4})$	$\tilde{O}(n^{-1/2})$


Key idea: Compress non-uniformly for high quality

Compress $\mathbf{X}_n$ to $\hat{\mathbf{X}}_m$ of size $m = \sqrt{n}$ with small $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m)$



KT-Compress = Kernel thinning (KT) + Compress

KT-Compress = Kernel thinning (KT) + Compress

KT: Take n^2 time to output m points with

$$\mathrm{MMD}_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m) = \tilde{O}(1/m)$$

KT-Compress = Kernel thinning (KT) + Compress

KT: Take n^2 time to output m points with

$$\mathrm{MMD}_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m) = \tilde{O}(1/m)$$

Works with an <u>arbitrary</u> sequence of points (IID, MCMC, …) and an <u>arbitrary</u> kernel (Gaussian, Matérn, …)

KT-Compress = Kernel thinning (KT) + Compress

KT: Take n^2 time to output m points with

$$\mathrm{MMD}_{\mathbf{k}}(\mathbf{X}_n, \hat{\mathbf{X}}_m) = \tilde{O}(1/m)$$

Works with an <u>arbitrary</u> sequence of points (IID, MCMC, …) and an <u>arbitrary</u> kernel (Gaussian, Matérn, …)

Kernel thinning, [Dwivedi & Mackey '21]

Generalized Kernel thinning, [Dwivedi & Mackey '22]

KT-Compress = Kernel thinning (KT) + Compress

KT-Compress = Kernel thinning (KT) + Compress

Compress improves KT's runtime from $\Theta(n^2)$ to $\tilde{O}(n)$ while making MMD error worse **by at most a factor of 4!**

Also, Compress provides a quadratic speedup to the runtime of generic thinning algorithms!

KT-Compress = Kernel thinning (KT) + Compress

Compress improves KT's runtime from $\Theta(n^2)$ to $\tilde{O}(n)$ while making MMD error worse **by at most a factor of 4!**

Also, Compress provides a quadratic speedup to the runtime of generic thinning algorithms!

Distribution compression in near-linear time,
[Shetty, Dwivedi & Mackey '22]

Kernel thinning: Algorithm overview

Recursive kernel halving:

$$n \to \frac{n}{2} \to \frac{n}{4} \to \cdots \to \sqrt{n}$$

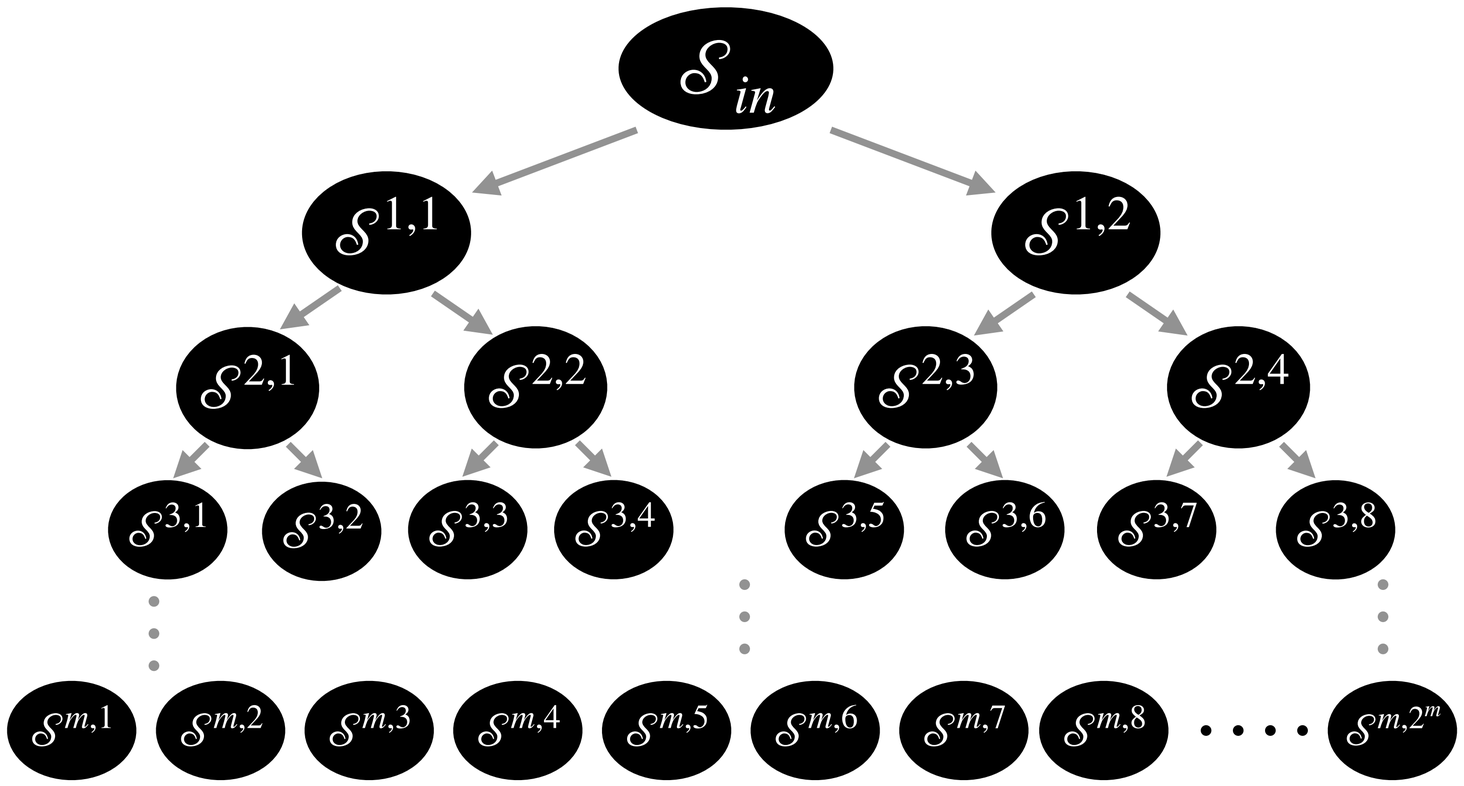
+ Post-processing step

Kernel thinning: Algorithm overview

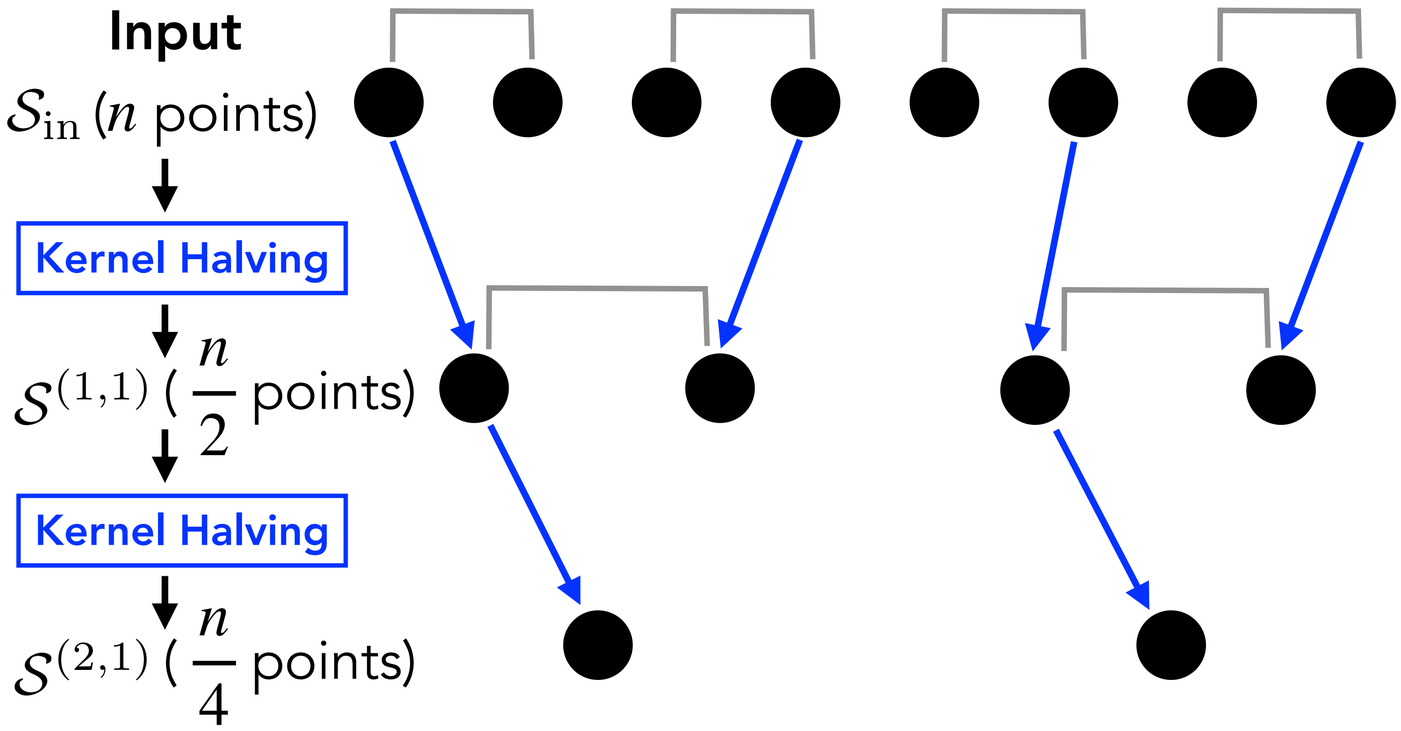
Recursive kernel halving:

$$n \to \frac{n}{2} \to \frac{n}{4} \to \cdots \to \sqrt{n}$$

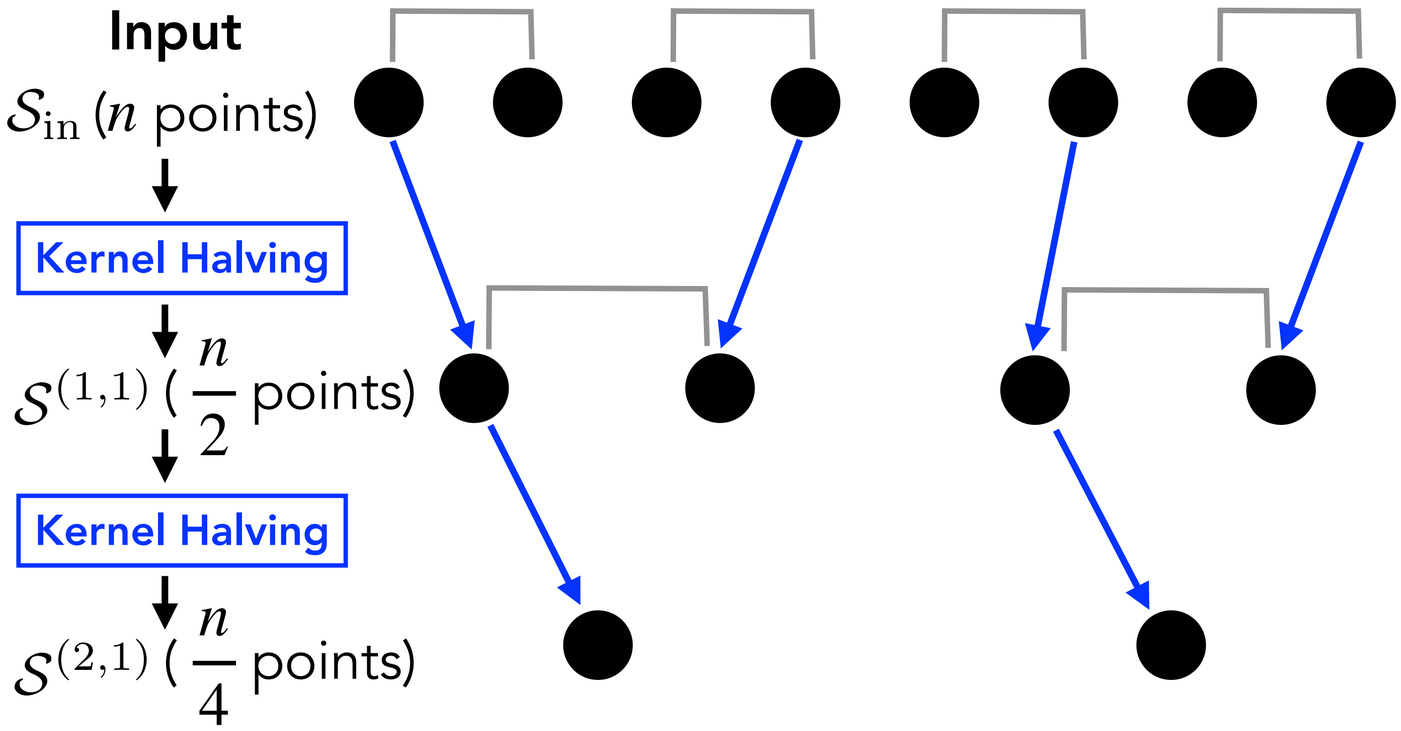
+ Post-processing step



Kernel halving = Sequential non-uniform selection



Kernel halving = Sequential non-uniform selection



Sequential selection of a point out of paired points, using biased coin flips toward the point with smaller MMD error to input points

Compress: Recursive strategy to reduce runtime

Thinning algorithm with n^2 runtime and error $e(n)$ → Compress → $\sqrt{n}$-thinning algorithm with $n \log^3 n$ runtime and error $4e(n)$

Algorithm 1: COMPRESS

Input: halving algorithm HALVE, oversampling parameter g, point sequence $\mathcal{S}_{\text{in}}$ of size n

if $n = 4^g$ **then return** $\mathcal{S}_{\text{in}}$

else

  Partition $\mathcal{S}_{\text{in}}$ into four arbitrary subsequences $\{\mathcal{S}_i\}_{i=1}^4$ each of size $n/4$

  for $i = 1, 2, 3, 4$ **do**

    $\widetilde{\mathcal{S}}_i \leftarrow$ COMPRESS$(\mathcal{S}_i, \text{HALVE}, g)$ // return coresets of size $2^g \cdot \sqrt{\frac{n}{4}}$

  end

  $\widetilde{\mathcal{S}} \leftarrow$ CONCATENATE$(\widetilde{\mathcal{S}}_1, \widetilde{\mathcal{S}}_2, \widetilde{\mathcal{S}}_3, \widetilde{\mathcal{S}}_4)$ // coreset of size $2 \cdot 2^g \cdot \sqrt{n}$

  return HALVE$(\widetilde{\mathcal{S}})$ // coreset of size $2^g\sqrt{n}$

end

CTT: Compress (the data and) then test

CTT: Compress (the data and) then test

1. **KT-Compress:** $\mathbf{X}_n \to \hat{\mathbf{X}}_m$ & $\mathbf{Y}_n \to \hat{\mathbf{Y}}_m$ with size $m = 2^{\mathfrak{g}}\sqrt{n}$

CTT: Compress (the data and) then test

1. **KT-Compress:** $\mathbf{X}_n \to \hat{\mathbf{X}}_m$ & $\mathbf{Y}_n \to \hat{\mathbf{Y}}_m$ with size $m = 2^{\mathfrak{g}}\sqrt{n}$

2. **Test statistic:** $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \to \mathrm{MMD}^2_{\mathbf{k}}(\hat{\mathbf{X}}_m, \hat{\mathbf{Y}}_m)$

CTT: Compress (the data and) then test

1. **KT-Compress:** $\mathbf{X}_n \to \hat{\mathbf{X}}_m$ & $\mathbf{Y}_n \to \hat{\mathbf{Y}}_m$ with size $m = 2^{\mathfrak{g}}\sqrt{n}$

2. **Test statistic:** $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \to \mathrm{MMD}^2_{\mathbf{k}}(\hat{\mathbf{X}}_m, \hat{\mathbf{Y}}_m)$

3. **Threshold:** Via *cheap permutations*, group data into $s \ll \sqrt{n}$ bins, sample $\mathscr{L}$ permutations of $[s]$, and permute the s bins

CTT: Compress (the data and) then test

1. **KT-Compress:** $\mathbf{X}_n \to \hat{\mathbf{X}}_m$ & $\mathbf{Y}_n \to \hat{\mathbf{Y}}_m$ with size $m = 2^{\mathfrak{g}}\sqrt{n}$

2. **Test statistic:** $\mathrm{MMD}^2_{\mathbf{k}}(\mathbf{X}_n, \mathbf{Y}_n) \to \mathrm{MMD}^2_{\mathbf{k}}(\hat{\mathbf{X}}_m, \hat{\mathbf{Y}}_m)$

3. **Threshold:** Via *cheap permutations*, group data into $s \ll \sqrt{n}$ bins, sample $\mathscr{L}$ permutations of $[s]$, and permute the s bins

CTT runtime: $\tilde{O}(n) + O(s^2\mathscr{L})$ if $\mathfrak{g} = \log\log n$

Complete MMD test runtime: $O(n^2) + O(n^2\mathscr{L})$

Test name	MMD separation	Runtime
CTT (ours)	$\frac{\mathsf{R}_{\mathbf{k},\mathbf{k}'}(\mathbb{P}, n, \delta, \mathfrak{g})}{2^{\mathfrak{g}}\sqrt{n}} + n^{-\frac{1}{2}}$	$4^{\mathfrak{g}} n \log n$
Complete MMD (Gretton et al., 2012a)	$n^{-\frac{1}{2}}$	n^2
Block MMD (Zaremba et al., 2013)	$(Bn)^{-\frac{1}{4}}$	Bn
Incomplete MMD (Yamada et al., 2019)	$\ell^{-\frac{1}{4}}$	ℓ

B = Number of blocks used in Block MMD test

ℓ = Number of ordered index pairs in Incomplete MMD test

δ = Failure probability for CTT guarantees

$\mathbf{k}'$ = auxiliary kernel used by KT-Compress

Test name	MMD separation	Runtime
CTT (ours)	$\frac{\mathsf{R}_{\mathbf{k},\mathbf{k}'}(\mathbb{P}, n, \delta, \mathfrak{g})}{2^{\mathfrak{g}}\sqrt{n}} + n^{-\frac{1}{2}}$	$4^{\mathfrak{g}} n \log n$
Complete MMD (Gretton et al., 2012a)	$n^{-\frac{1}{2}}$	n^2
Block MMD (Zaremba et al., 2013)	$(Bn)^{-\frac{1}{4}}$	Bn
Incomplete MMD (Yamada et al., 2019)	$\ell^{-\frac{1}{4}}$	ℓ

Tails of $\mathbb{P}$ on $\mathbb{R}^d$	Choice of $\mathbf{k}'$	$\mathsf{R}_{\mathbf{k},\mathbf{k}'}(\mathbb{P}, n, \delta, \mathfrak{g})$
Compact	Compact $\mathbf{k}_{\mathrm{rt}}$	$\left(\log \frac{n}{\delta}\right)^2$
Subexp	Analytic $\mathbf{k}$	$\left(\log \frac{n}{\delta}\right)^{\frac{3d+5}{2}}$
Subexp	Subexp $\mathbf{k}_{\mathrm{rt}}$	$c_{m,\delta}\left(\log \frac{n}{\delta}\right)^{\frac{d+5}{2}}$
ρ-Heavy-tail	ρ-Heavy-tail $\mathbf{k}_{\mathrm{rt}}$	$\left(\frac{n}{\delta}\right)^{\frac{d}{2\rho}}\left(\log \frac{n}{\delta}\right)^{\frac{d+5}{2}}$

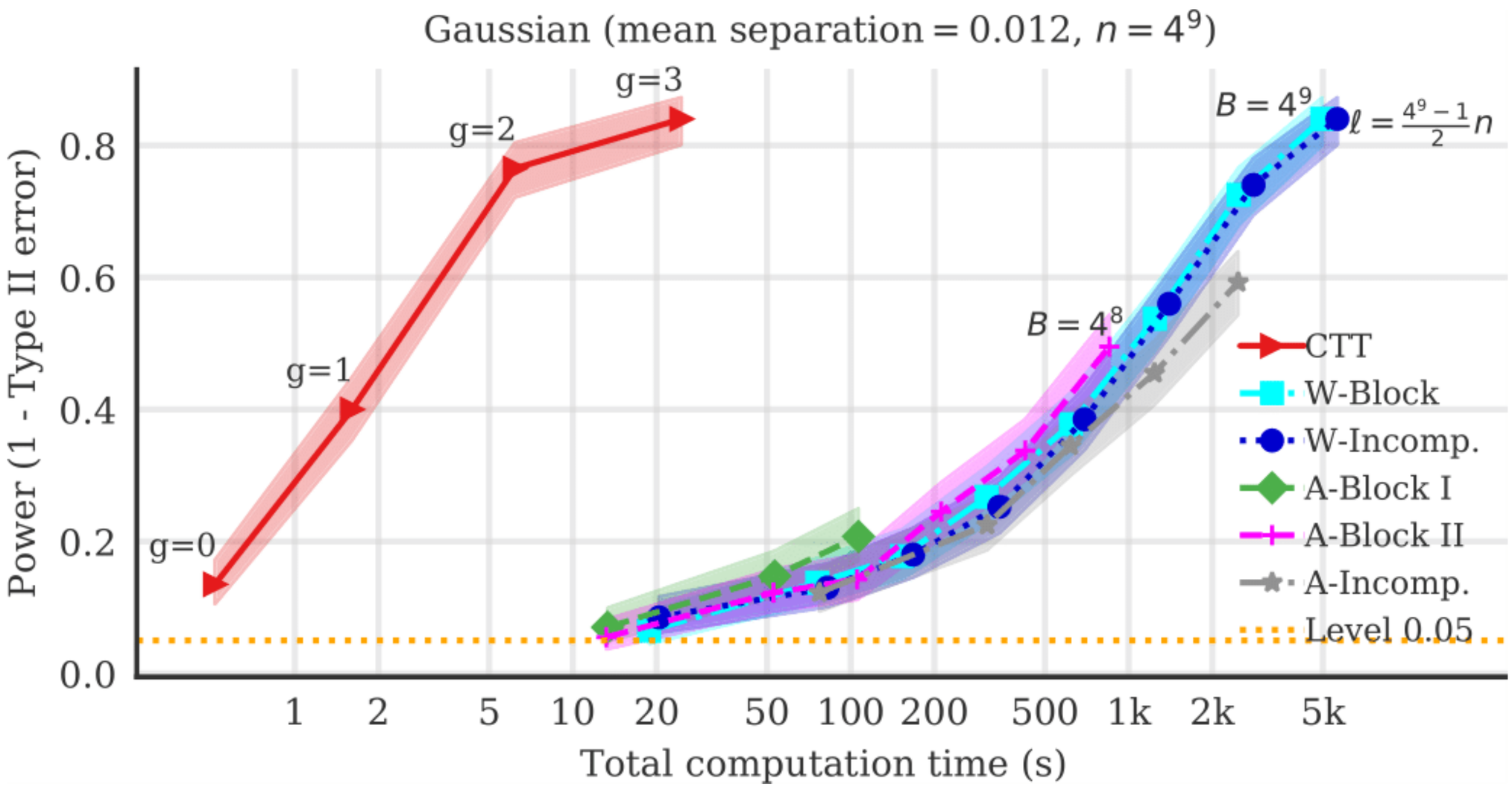
B = Number of blocks used in Block MMD test

ℓ = Number of ordered index pairs in Incomplete MMD test

δ = Failure probability for CTT guarantees

$\mathbf{k}'$ = auxiliary kernel used by KT-Compress & $\mathbf{k}(x, y) = \int \mathbf{k}_{\mathrm{rt}}(x, z)\mathbf{k}_{\mathrm{rt}}(z, y)dz$

Time-power tradeoff on Gaussian data



Compress Then Test: Powerful Kernel Testing in Near-linear Time [Domingo-Enrich, Dwivedi & Mackey '23]

1. Quadratic → Near-linear time tests using KT-Compress
2. Provide up to 200x speed-up (1 hour → 20 sec)
3. Maintain level *and* power provably
4. Work with kernel approximations (Low-rank CTT)
5. Applicable for kernel selection (Aggregated CTT)

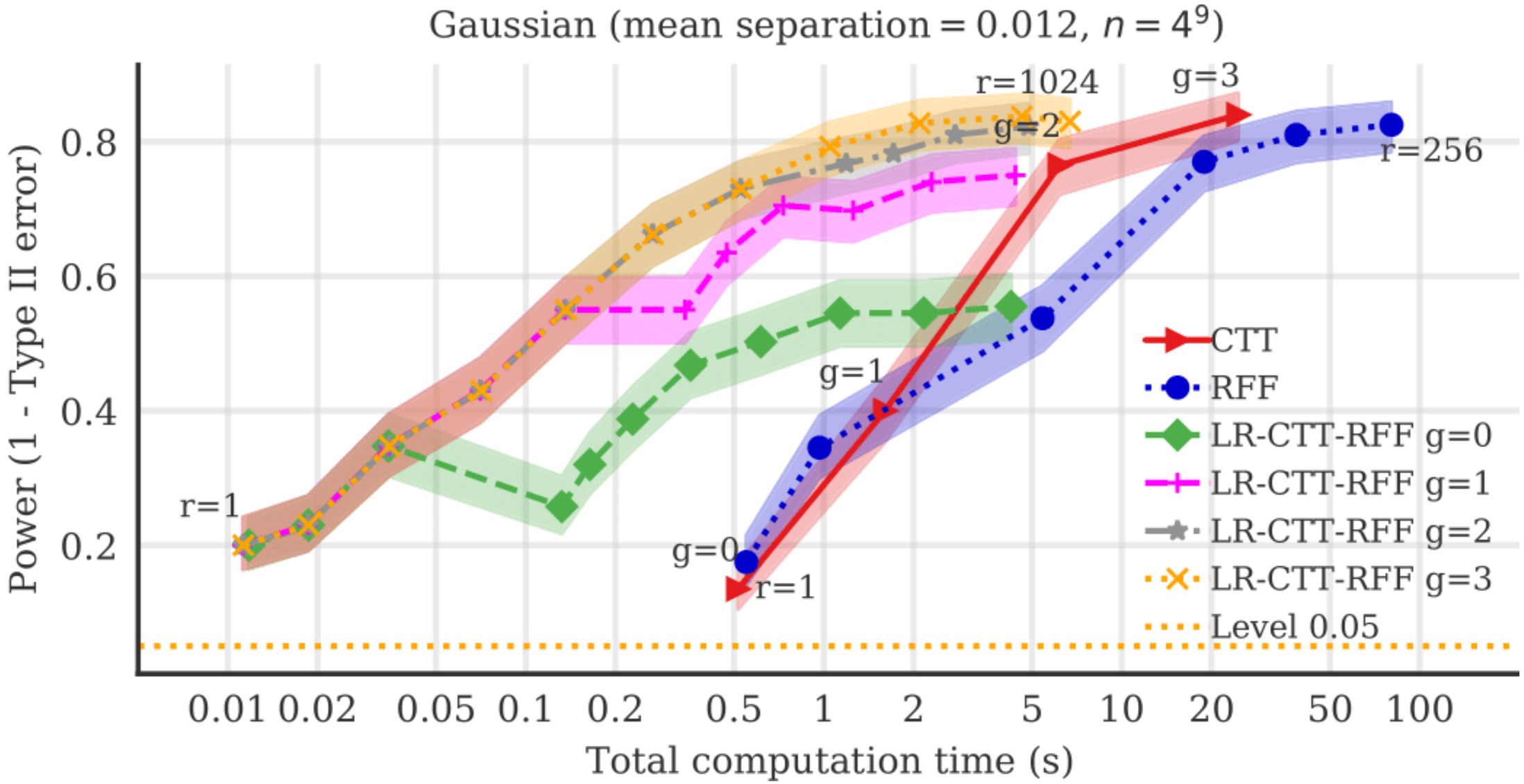
Goodpoints Python Library

github.com/microsoft/goodpoints

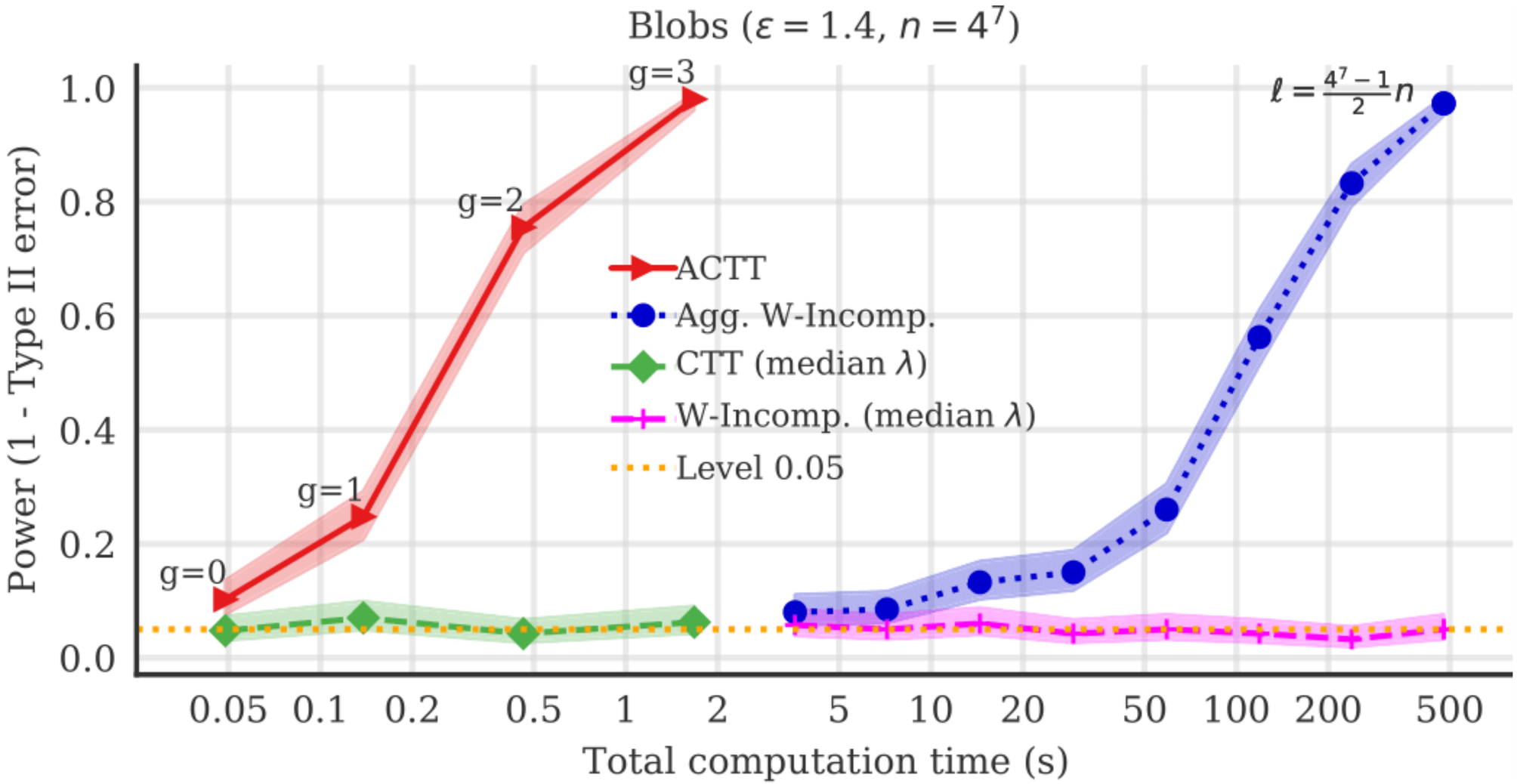
Kernel Thinning (KT)	n points $\rightarrow \sqrt{n}$ points with $\tilde{O}(n^{-1/2})$ error in n^2 time
KT-Compress	$\tilde{O}(n)$ runtime version of KT with error worse by a factor of 4
Compress then Test (CTT)	Powerful $\tilde{O}(n)$ runtime kernel tests using KT-Compress

Appendix

Low-rank CTT vs random Fourier features based test



Aggregated CTT for kernel selection



n points → **Standard Thinning** → $\sqrt{n}$ points with $\Theta(n^{-1/4})$ error (Monte Carlo rate)

n points → **Kernel Thinning (KT)**[1,2] → $\sqrt{n}$ points with $\tilde{O}(n^{-1/2})$ MMD error in n^2 time

n points → **KT-Compress++**[3] → $\sqrt{n}$ points with $\tilde{O}(n^{-1/2})$ MMD error in $n \log^3 n$ time

Two sets of n points → **Compress Then Test (CTT)**[4] → Powerful always valid two-sample test in $\tilde{O}(n)$ time

pip install goodpoints


Abhishek Shetty


Carles Domingo Enrich


Lester Mackey

1. Kernel Thinning, COLT 2021
2. Generalized Kernel Thinning, ICLR 2022
3. Distribution Compression In Near-linear Time, ICLR 2022
4. Compress Then Test: Powerful Kernel Testing in Near-Linear Time, AISTATS 2023

Compress++: Recursive strategy to reduce runtime

Thinning algorithm with n^2 runtime and error $e(n)$ → **Compress++** → $\sqrt{n}$-thinning algorithm with $n \log^3 n$ runtime and error $4e(n)$

Algorithm 1: COMPRESS

Input: halving algorithm HALVE, oversampling parameter $\mathfrak{g}$, point sequence $\mathcal{S}_{\text{in}}$ of size n

if $n = 4^{\mathfrak{g}}$ **then return** $\mathcal{S}_{\text{in}}$

else

  Partition $\mathcal{S}_{\text{in}}$ into four arbitrary subsequences $\{\mathcal{S}_i\}_{i=1}^{4}$ each of size $n/4$

  for $i = 1, 2, 3, 4$ **do**

    $\widetilde{\mathcal{S}}_i \leftarrow$ COMPRESS$(\mathcal{S}_i, \text{HALVE}, \mathfrak{g})$ // return coresets of size $2^{\mathfrak{g}} \cdot \sqrt{\frac{n}{4}}$

  end

  $\widetilde{\mathcal{S}} \leftarrow$ CONCATENATE$(\widetilde{\mathcal{S}}_1, \widetilde{\mathcal{S}}_2, \widetilde{\mathcal{S}}_3, \widetilde{\mathcal{S}}_4)$ // coreset of size $2 \cdot 2^{\mathfrak{g}} \cdot \sqrt{n}$

  return HALVE$(\widetilde{\mathcal{S}})$ // coreset of size $2^{\mathfrak{g}}\sqrt{n}$

end