Transfer Learning for Optimal Configuration of Big Data Software

Pooyan Jamshidi, Giuliano Casale
Imperial College London
p.jamshidi@imperial.ac.uk

Department of Computing
Research Associate Symposium
14th June 2016

Imperial College London

Motivation

1- *Many different Parameters =>*
- *large state space*
- *interactions*

2- *Defaults are typically used =>*
- *poor performance*

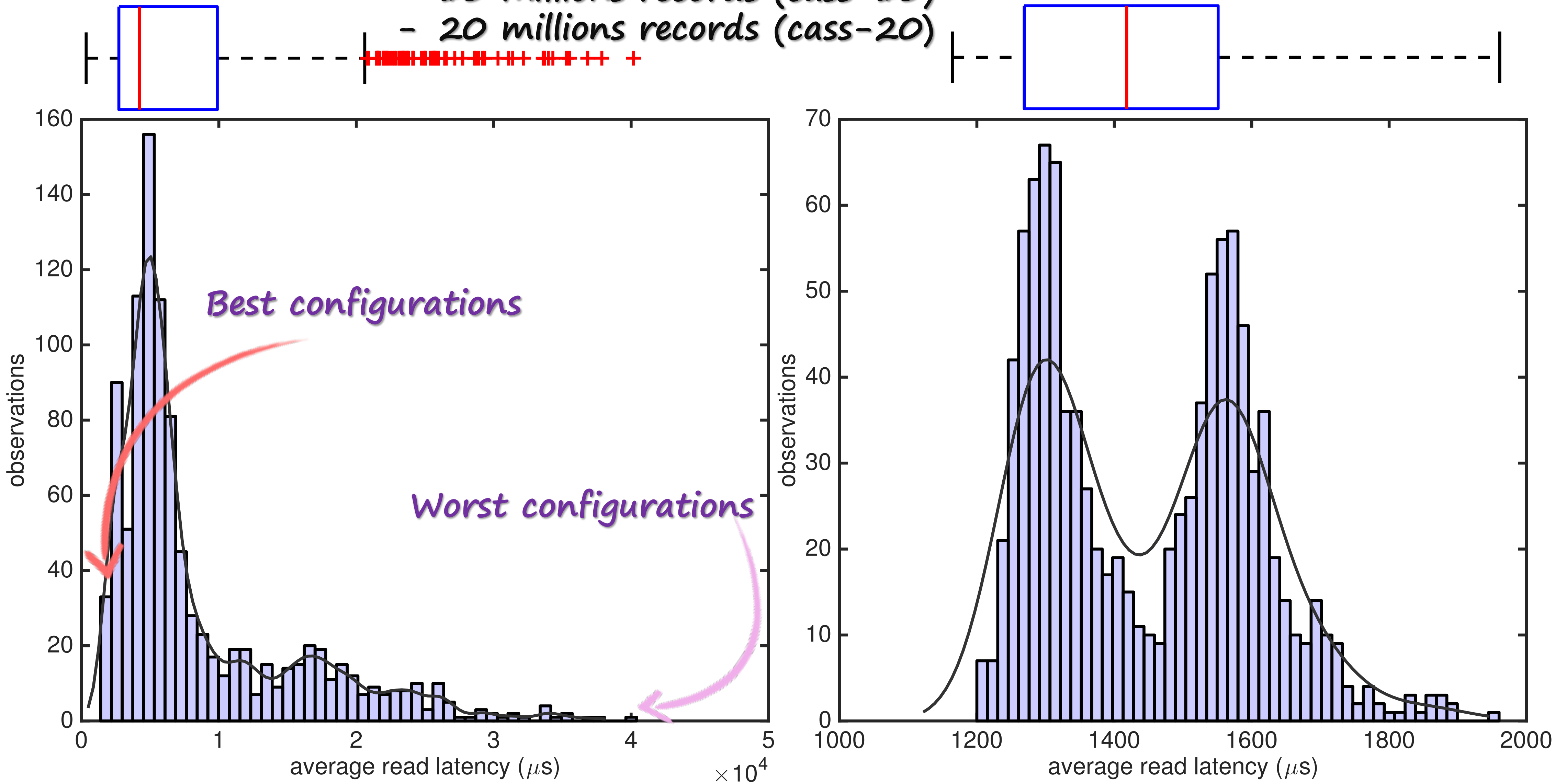
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

Motivation

Experiments on
Apache Cassandra:
- 6 parameters, 1024 configurations
- Average read latency
- 10 millions records (cass-10)
- 20 millions records (cass-20)



(a) cass-20

(b) cass-10

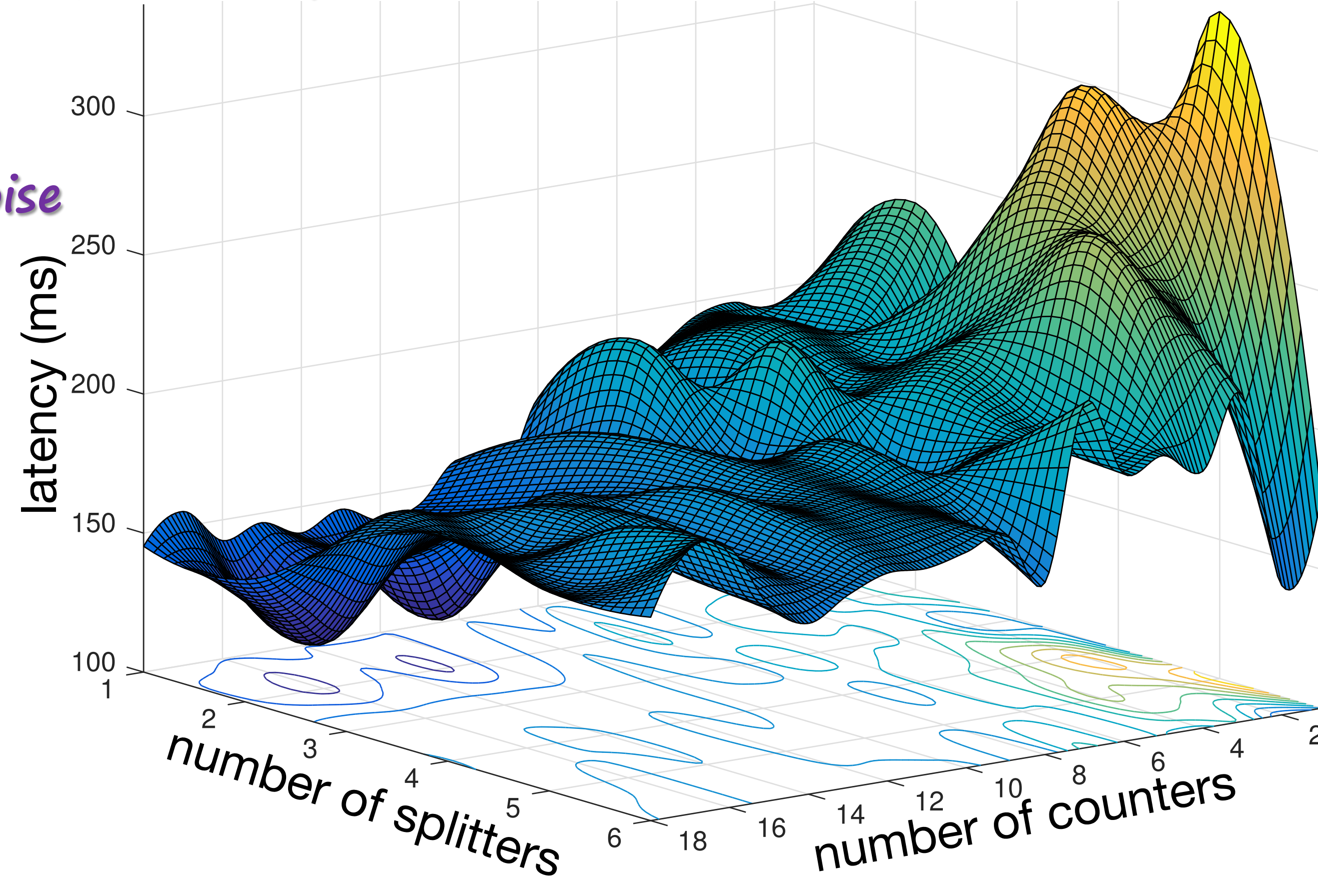
Goal!

$$x^* = \arg\min_{x \in \mathbb{X}} f(x)$$

$$\mathbb{X} = Dom(X_1) \times \cdots \times Dom(X_d)$$ Configuration space

$$y_i = f(x_i), x_i \subset \mathbb{X}$$ Partially known

$$y_i = f(x_i) + \epsilon$$ Measurements contain noise



Finding optimum configuration is difficult!

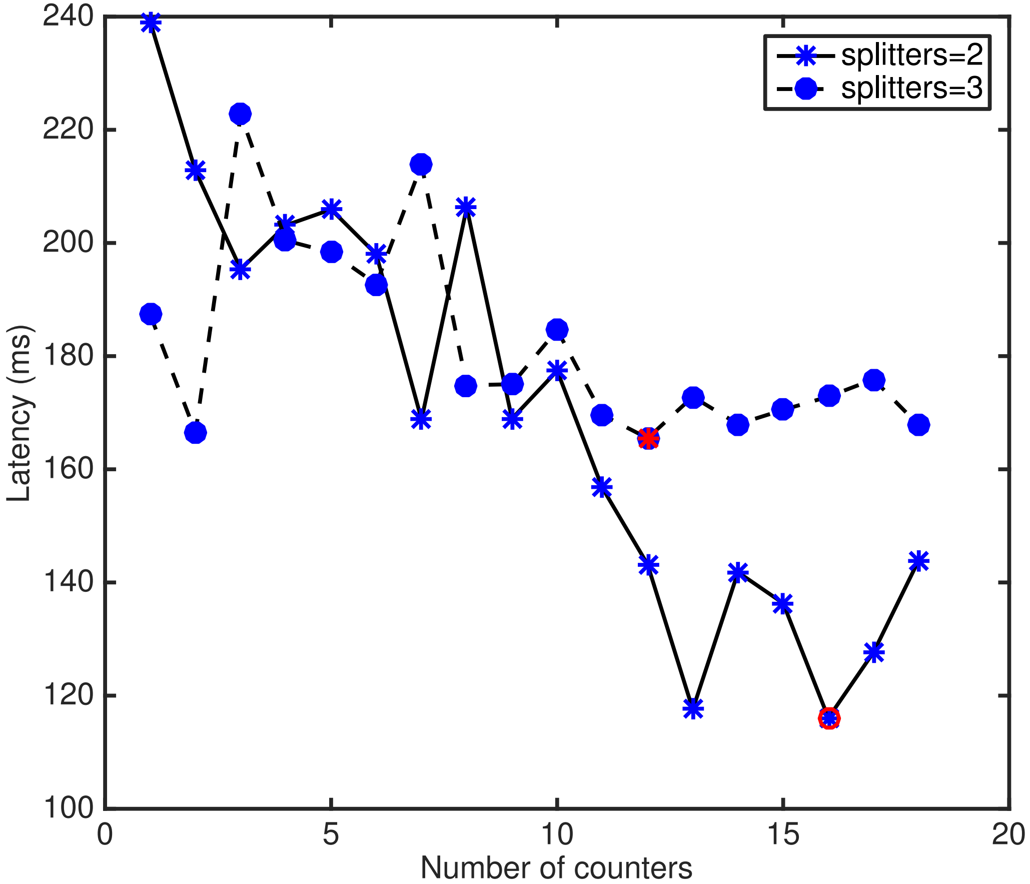
$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{X}} f(\boldsymbol{x})$$

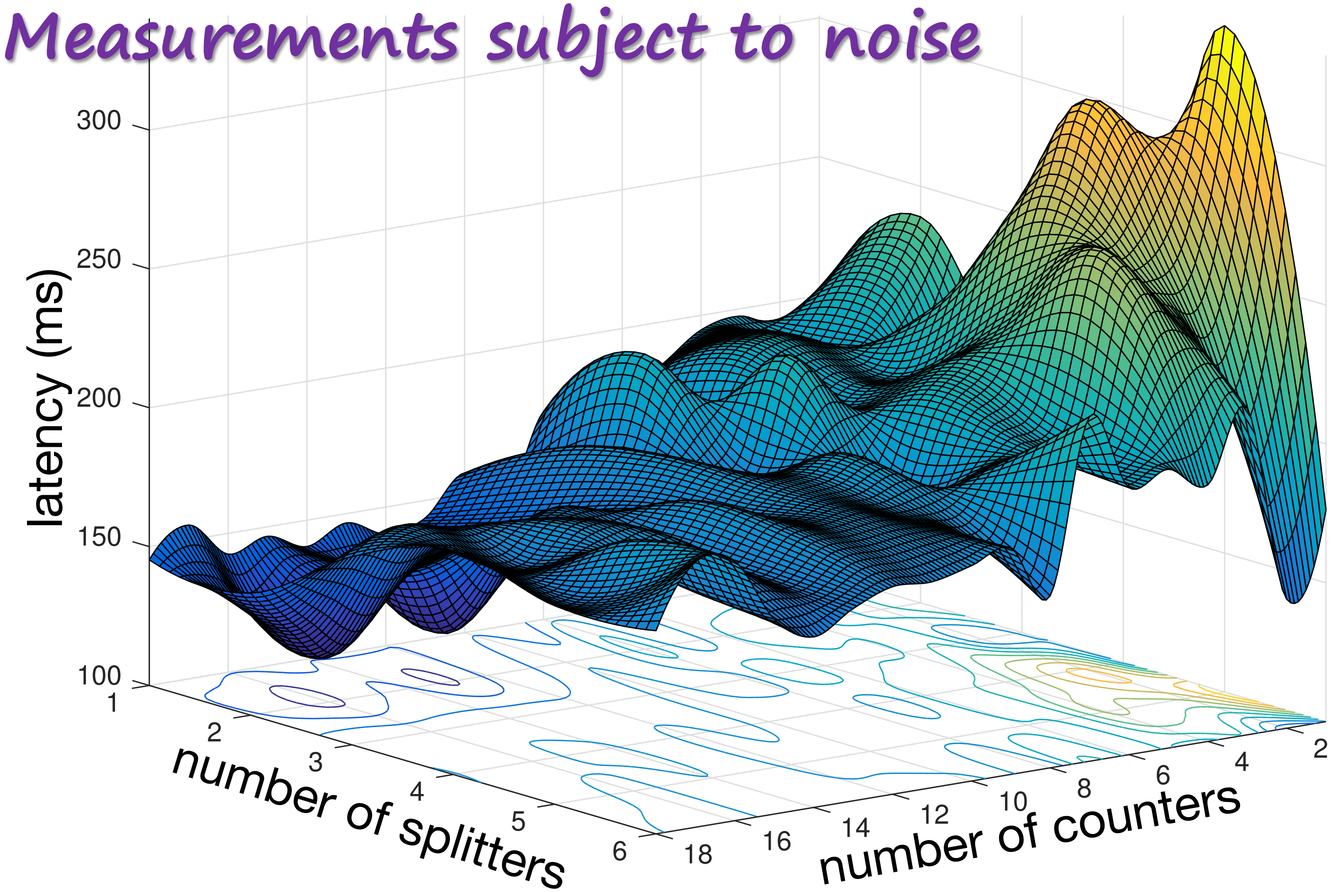
$$\mathbb{X} = Dom(X_1) \times \cdots \times Dom(X_d)$$

$$y_i = f(\boldsymbol{x}_i), \boldsymbol{x}_i \subset \mathbb{X}$$

$$y_i = f(\boldsymbol{x}_i) + \epsilon$$

Response surface is:
- Non-linear
- Non convex
- Multi-modal
- Measurements subject to noise





Part

Bayesian Optimization for Configuration Optimization (BO4CO)

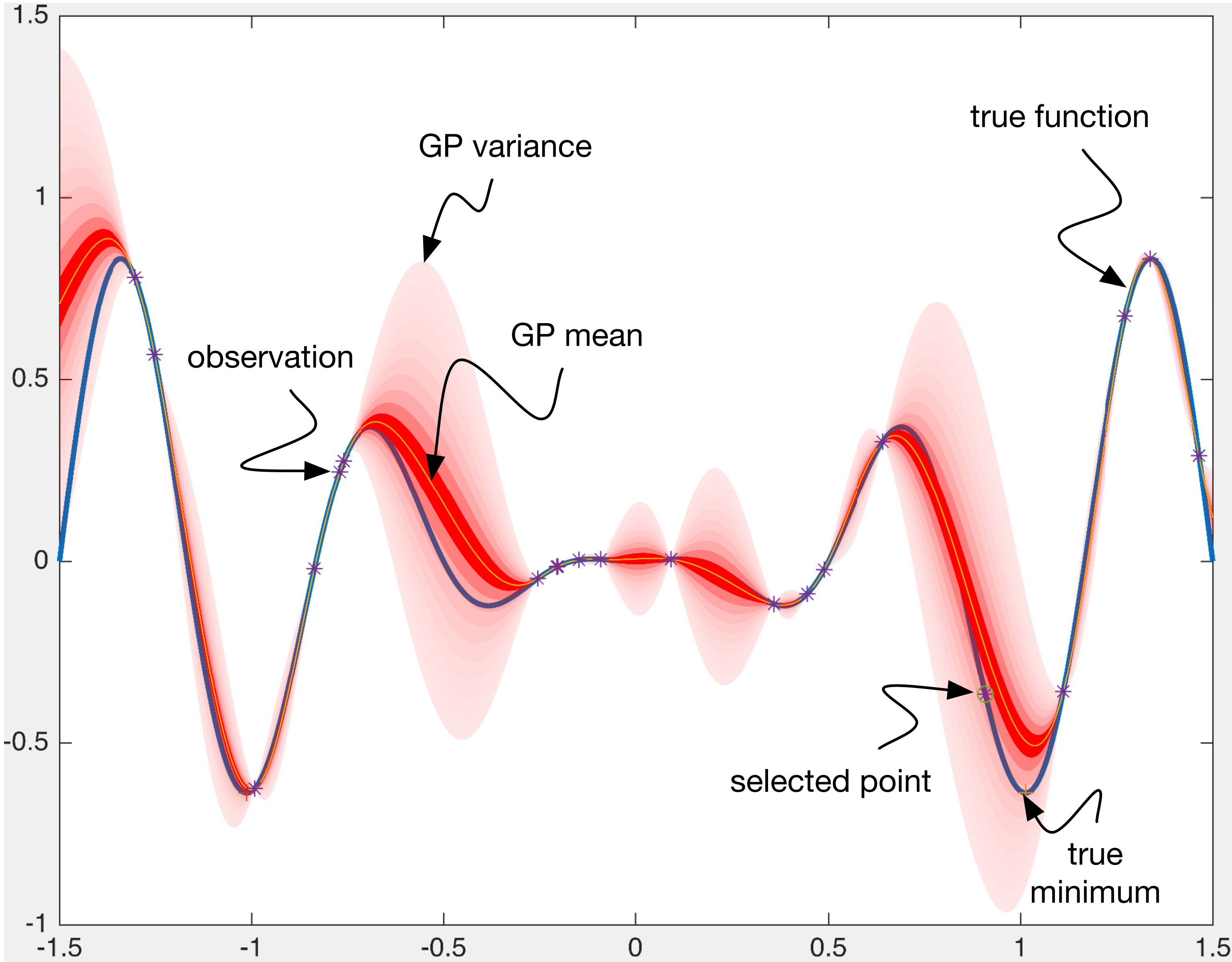
Code: https://github.com/dice-project/DICE-Configuration-BO4CO

GP for modeling blackbox response function

$$y = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\mu_t(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^\intercal(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$$
$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2\boldsymbol{I} - \boldsymbol{k}(\boldsymbol{x})^\intercal(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{x})$$

Motivations:
1- mean estimates + variance
2- all computations are linear algebra



Kernel function:

$$\boldsymbol{K} := \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix}$$

$$k_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\Sigma_{\ell=1}^{d}(-\theta_\ell \delta(\boldsymbol{x}_i \neq \boldsymbol{x}_j))),$$

Acquisition function:

$$u_{LCB}(\boldsymbol{x}|\mathcal{M}, \mathbb{S}_{1:n}) = \underset{\boldsymbol{x} \in \mathbb{X}}{\operatorname{argmin}}\, \mu_t(\boldsymbol{x}) - \kappa \sigma_t(\boldsymbol{x}),$$

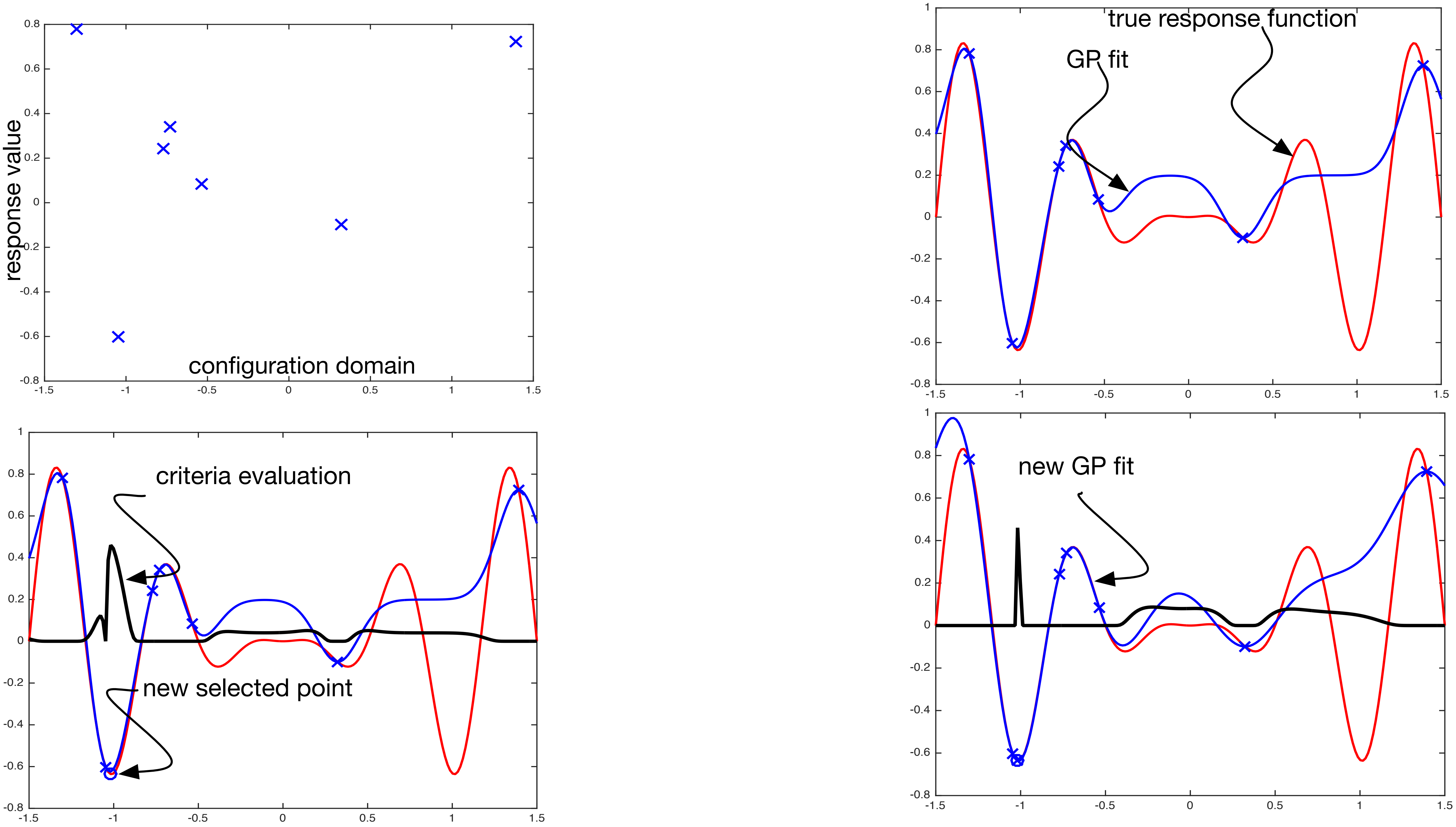
Code: https://github.com/pooyanjamshidi/BO4CO

Algorithm 1 : BO4CO

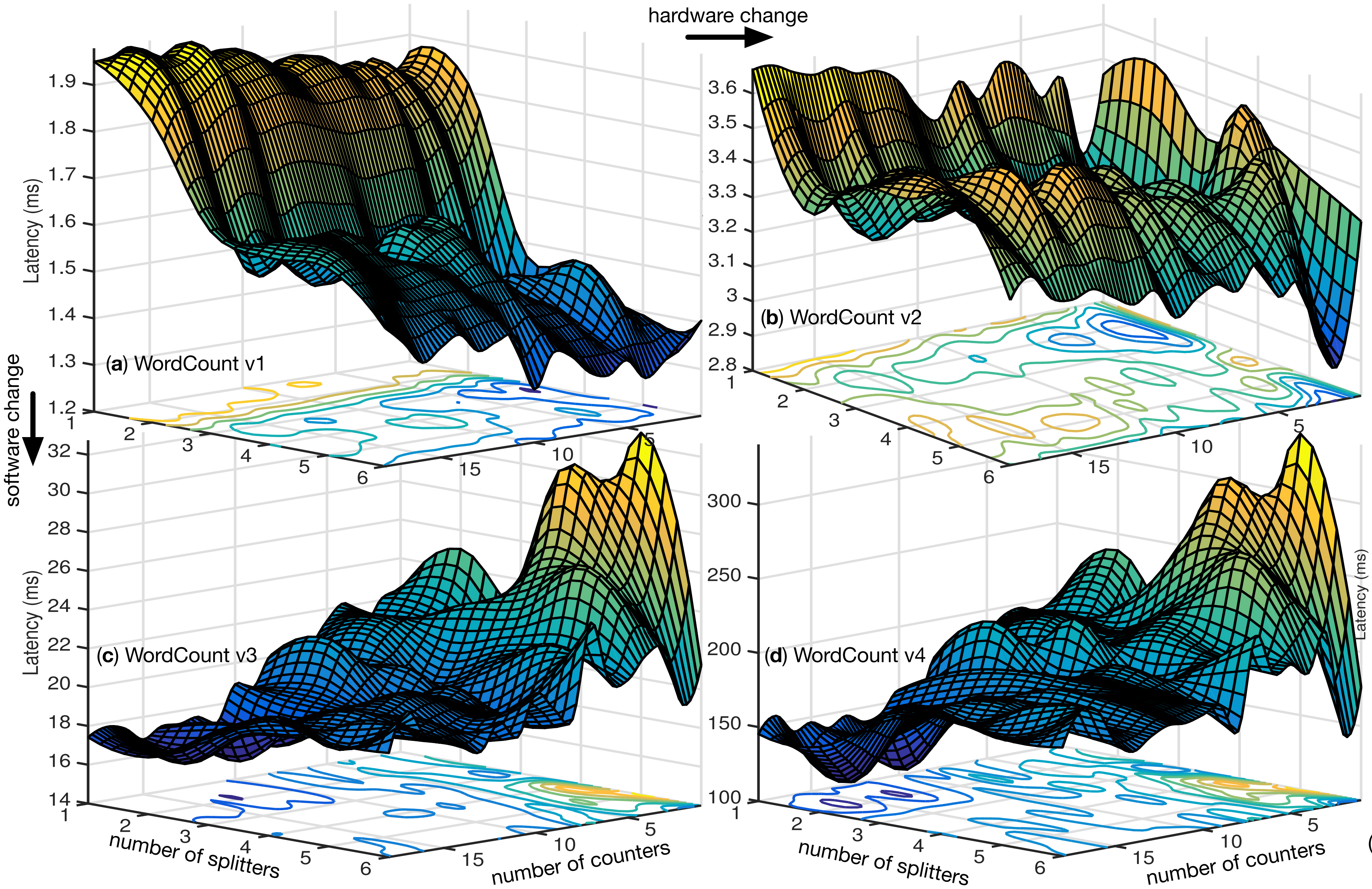
Input: Configuration space $\mathbb{X}$, Maximum budget N_{max}, Response function f, Kernel function $\boldsymbol{K}_\theta$, Hyper-parameters $\boldsymbol{\theta}$, Design sample size n, learning cycle N_l

Output: Optimal configurations $\boldsymbol{x}^*$ and learned model $\mathcal{M}$

choose an initial sparse design (*lhd*) to find an initial design samples $\mathcal{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$
obtain *performance measurements* of the initial design, $y_i \leftarrow f(\boldsymbol{x}_i) + \epsilon_i, \forall \boldsymbol{x}_i \in \mathcal{D}$
$\mathbb{S}_{1:n} \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}; t \leftarrow n + 1$
$\mathcal{M}(\boldsymbol{x}|\mathbb{S}_{1:n}, \boldsymbol{\theta}) \leftarrow$ fit a $\mathcal{GP}$ model to the design ▷ Eq.(3)
while $t \leq N_{max}$ **do**
  if $(t \mod N_l = 0)$ $\boldsymbol{\theta} \leftarrow$ *learn* the kernel hyper-parameters by maximizing the likelihood
  find *next configuration* $\boldsymbol{x}_t$ by optimizing the selection criteria over the estimated response surface given the data, $\boldsymbol{x}_t \leftarrow \arg max_{\boldsymbol{x}} u(\boldsymbol{x}|\mathcal{M}, \mathbb{S}_{1:t-1})$ ▷ Eq.(9)
  obtain performance for the *new configuration* $\boldsymbol{x}_t$, $y_t \leftarrow f(\boldsymbol{x}_t) + \epsilon_t$
  Augment the configuration $\mathbb{S}_{1:t} = \{\mathbb{S}_{1:t-1}, (\boldsymbol{x}_t, y_t)\}$
  $\mathcal{M}(\boldsymbol{x}|\mathbb{S}_{1:t}, \boldsymbol{\theta}) \leftarrow$ *re-fit* a new GP model ▷ Eq.(7)
  $t \leftarrow t + 1$
end while
$(\boldsymbol{x}^*, y^*) = \min \mathbb{S}_{1:N_{max}}$
$\mathcal{M}(\boldsymbol{x})$


response value
configuration domain
true response function
GP fit
criteria evaluation
new selected point
new GP fit

Correlations across different versions



correlation coefficient →

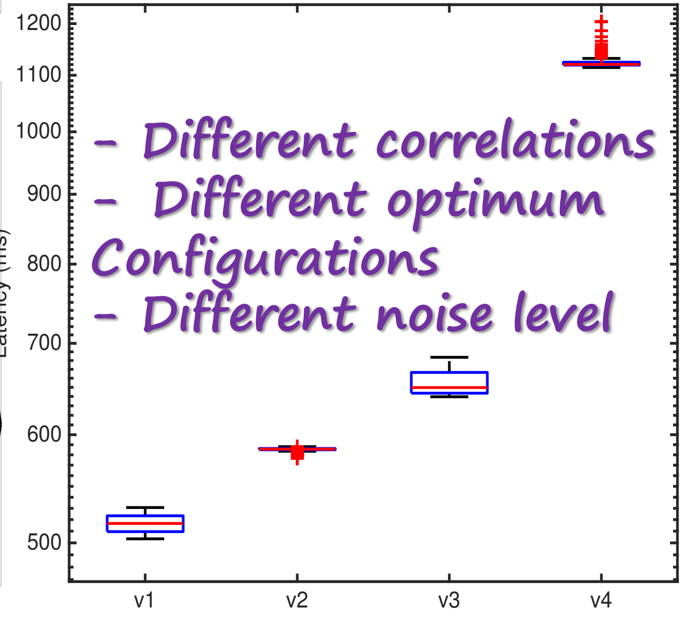
p-value ↓

	v1	v2	v3	v4
v1	1	0.41	-0.46	-0.50
v2	7.36E-06	1	-0.20	-0.18
v3	6.92E-07	0.04	1	0.94
v4	2.54E-08	0.07	1.16E-52	1

(**e**) Pearson correlation coefficients

	v1	v2	v3	v4
v1	1	0.49	-0.51	-0.51
v2	5.50E-08	1	-0.2793	-0.24
v3	1.30E-08	0.003	1	0.88
v4	1.40E-08	0.01	8.30E-36	1

(**f**) Spearman correlation coefficients



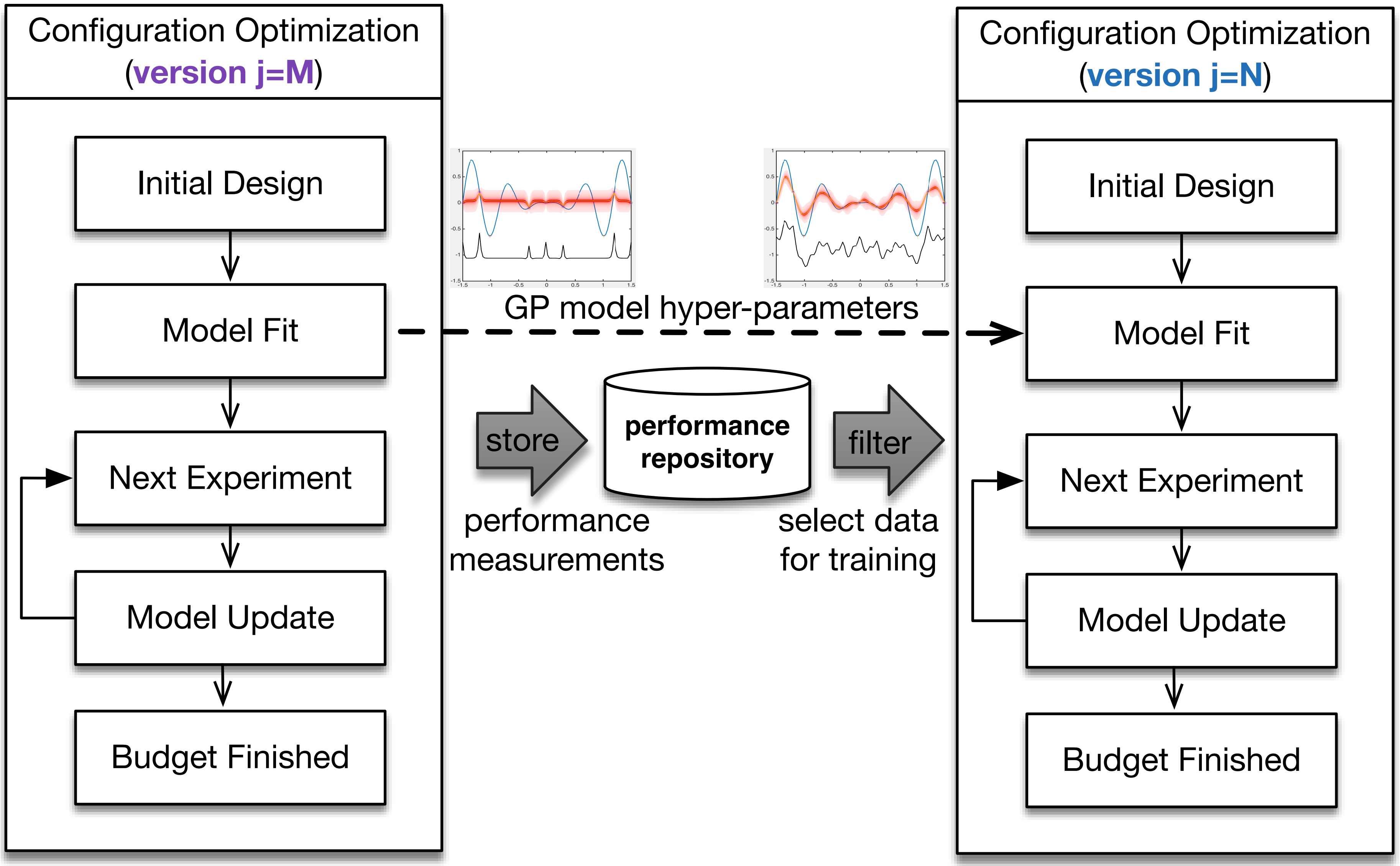
- Different correlations
- Different optimum Configurations
- Different noise level

(**g**) Measurement noise across WordCount versions

DevOps

- Different versions are continuously delivered (daily basis).
- Big Data systems are developed using similar frameworks (Apache Storm, Spark, Hadoop, Kafka, etc).
- Different versions share similar business logics.

Solution: Transfer Learning for Configuration Optimization



Part

Transfer Learning for Configuration Optimization (TL4CO)

Code: https://github.com/dice-project/DICE-Configuration-TL4CO

$$\mu_t(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}^\intercal(\boldsymbol{K}(X,X)+\Sigma)^{-1}(\boldsymbol{y}-\boldsymbol{\mu})$$

$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) + \sigma^2 - \boldsymbol{k}^\intercal(\boldsymbol{K}(X,X)+\Sigma)^{-1}\boldsymbol{k},$$

$$k_{TL4CO}(l,l',\boldsymbol{x},\boldsymbol{x}') = k_t(l,l') \times k_{xx}(\boldsymbol{x},\boldsymbol{x}'),$$

correlation between different versions

covariance functions

Code: https://github.com/pooyanjamshidi/TL4CO

Algorithm 1 : TL4CO

Input: Configuration space $\mathbb{X}$, Number of historical configuration optimization datasets T, Maximum budget N_{max}, Response function f, Kernel function $\boldsymbol{K}$, Hyper-parameters $\boldsymbol{\theta}^j$, The current dataset $\mathbb{S}^{j=1}$, Datasets belonging to other versions $\mathbb{S}^{j=2:T}$, Diagonal noise matrix Σ, Design sample size n, Learning cycle N_l

Output: Optimal configurations $\boldsymbol{x}^*$ and learned model $\mathcal{M}$

$\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ create an initial sparse design (*lhd*)

Obtain the *performance* $\forall x_i \in \mathcal{D}$, $y_i \leftarrow f(\boldsymbol{x}_i) + \epsilon_i$

$\mathbb{S}^{j=2:T} \leftarrow$ select m points from other versions ($j = 2 : T$) that reduce entropy ▷ Eq.(8)

$\mathbb{S}^1_{1:n} \leftarrow \mathbb{S}^{j=2:T} \cup \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$; $t \leftarrow n+1$

$\mathcal{M}(\boldsymbol{x}|\mathbb{S}^1_{1:n}, \boldsymbol{\theta^j}) \leftarrow$ fit a multi-task $\mathcal{GP}$ to $\mathcal{D}$ ▷ Eq. (6)

while $t \le N_{max}$ **do**

  If (t mod $N_l = 0$) then [$\boldsymbol{\theta} \leftarrow$ *learn* the kernel hyper-parameters by maximizing the likelihood]

  Determine the *next configuration*, $\boldsymbol{x}_t$, by optimizing LCB over $\mathcal{M}$, $\boldsymbol{x}_t \leftarrow \arg max_{\boldsymbol{x}} u(\boldsymbol{x}|\mathcal{M}, \mathbb{S}^1_{1:t-1})$ ▷ Eq.(11)

  Obtain the *performance* of $\boldsymbol{x}_t$, $y_t \leftarrow f(\boldsymbol{x}_t) + \epsilon_t$

  Augment the configuration $\mathbb{S}^1_{1:t} = \mathbb{S}^1_{1:t-1} \bigcup \{(\boldsymbol{x}_t, y_t)\}$

  $\mathcal{M}(\boldsymbol{x}|\mathbb{S}^1_{1:t}, \boldsymbol{\theta}) \leftarrow$ *re-fit* a new GP model ▷ Eq.(6)

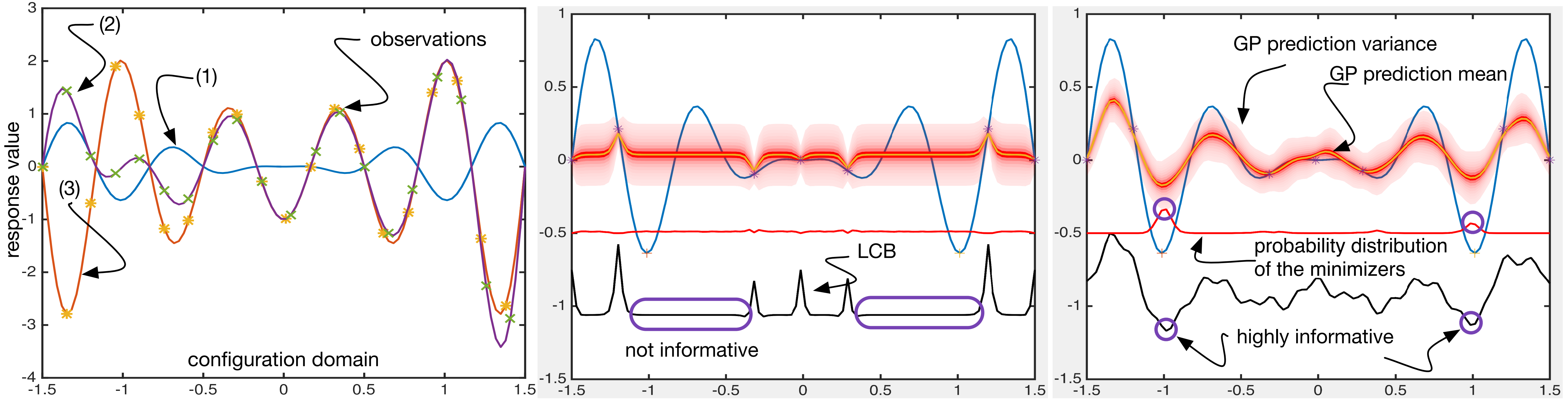
  $t \leftarrow t+1$

end while

$(\boldsymbol{x}^*, y^*) = \min[\mathbb{S}^1_{1:N_{max}} - \mathbb{S}^{j=2:T}]$ locating the optimum configuration by discarding data for other system versions

$\mathcal{M}(\boldsymbol{x})$ storing the learned model

Multi-task GP vs single-task GP



(a) 3 sample response functions

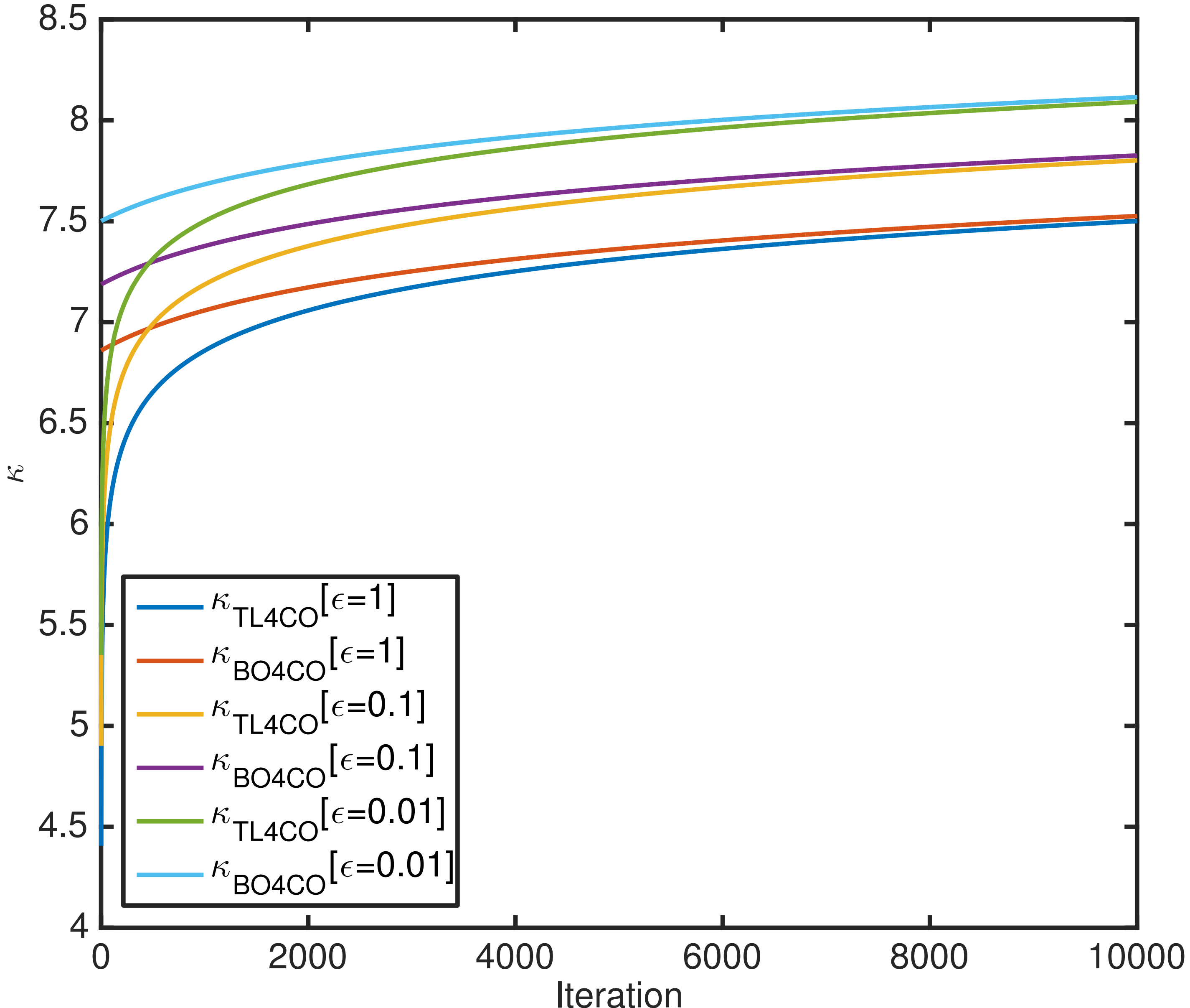
(b) GP fit for (1) ignoring observations for (2),(3)

(c) multi-task GP fit for (1) by transfer learning from (2),(3)

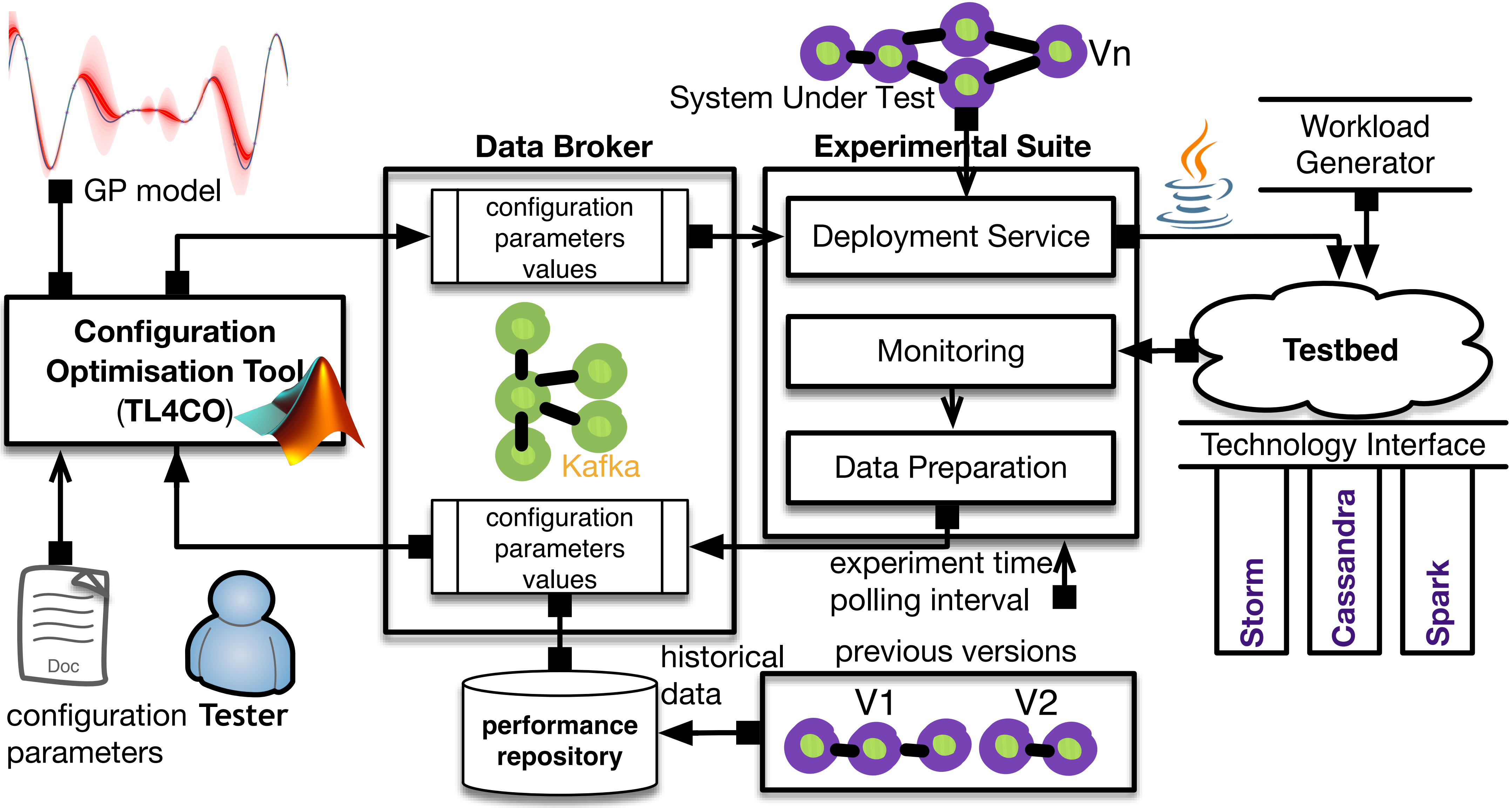
Exploitation vs exploration

$$\boldsymbol{x}_{t+1} = \underset{\boldsymbol{x} \in \mathbb{X}}{\operatorname{argmax}}\, u(\boldsymbol{x}|\mathcal{M}, \mathbb{S}^1_{1:t})$$

$$u_{LCB}(\boldsymbol{x}|\mathcal{M}, \mathbb{S}^1_{1:t}) = \underset{\boldsymbol{x} \in \mathbb{X}}{\operatorname{argmin}}\, \mu_t(\boldsymbol{x}) - \kappa\sigma_t(\boldsymbol{x}),$$

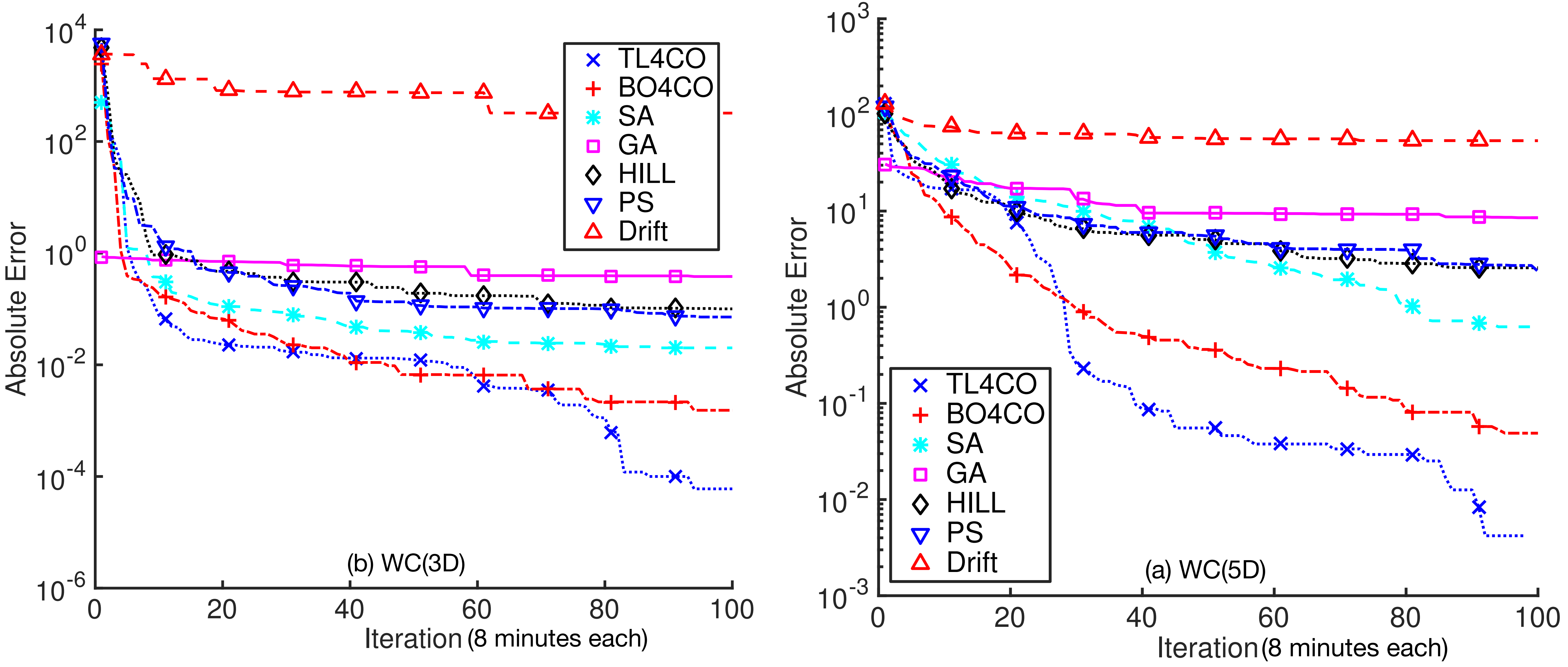


TL4CO architecture

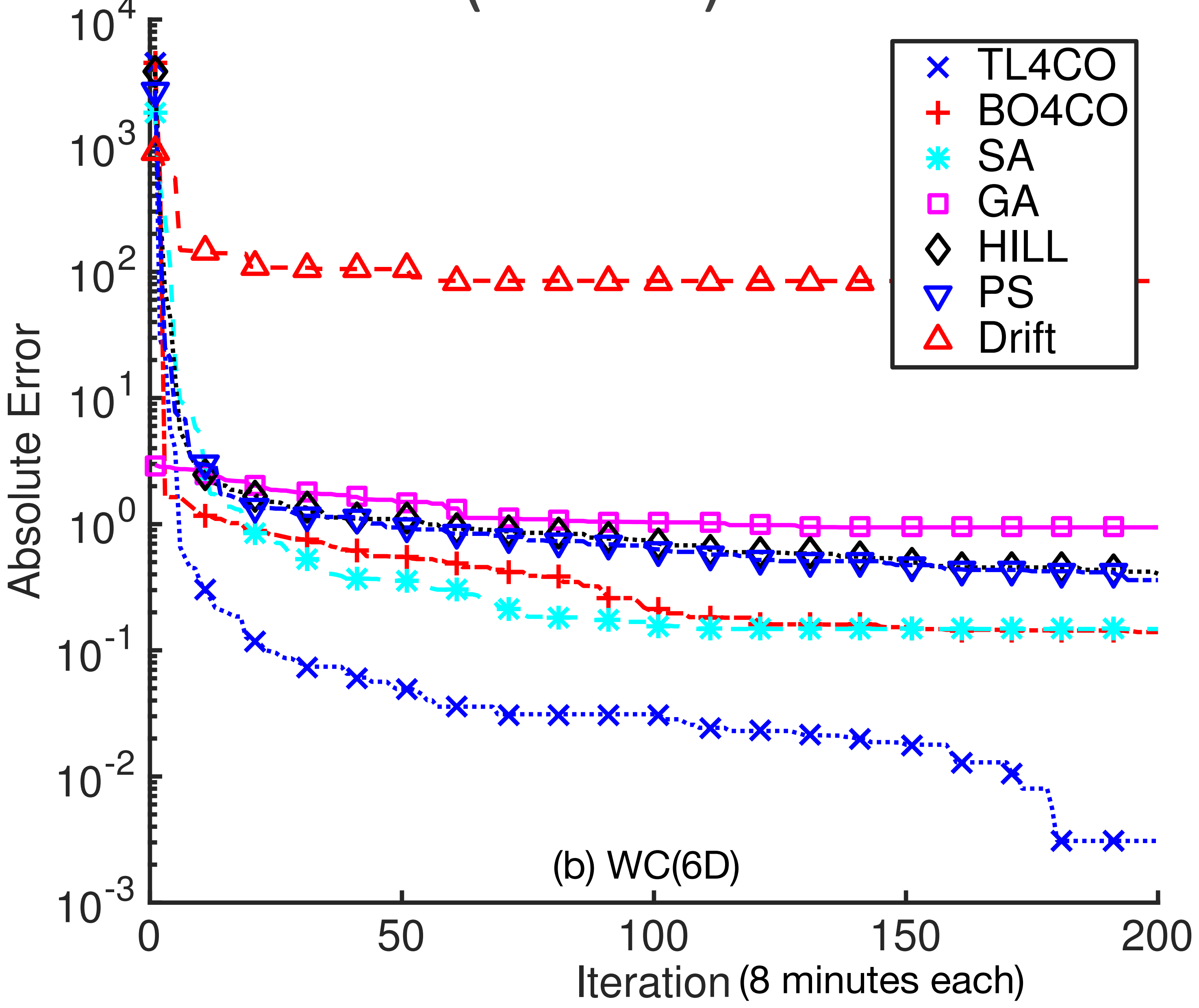


Experimental results

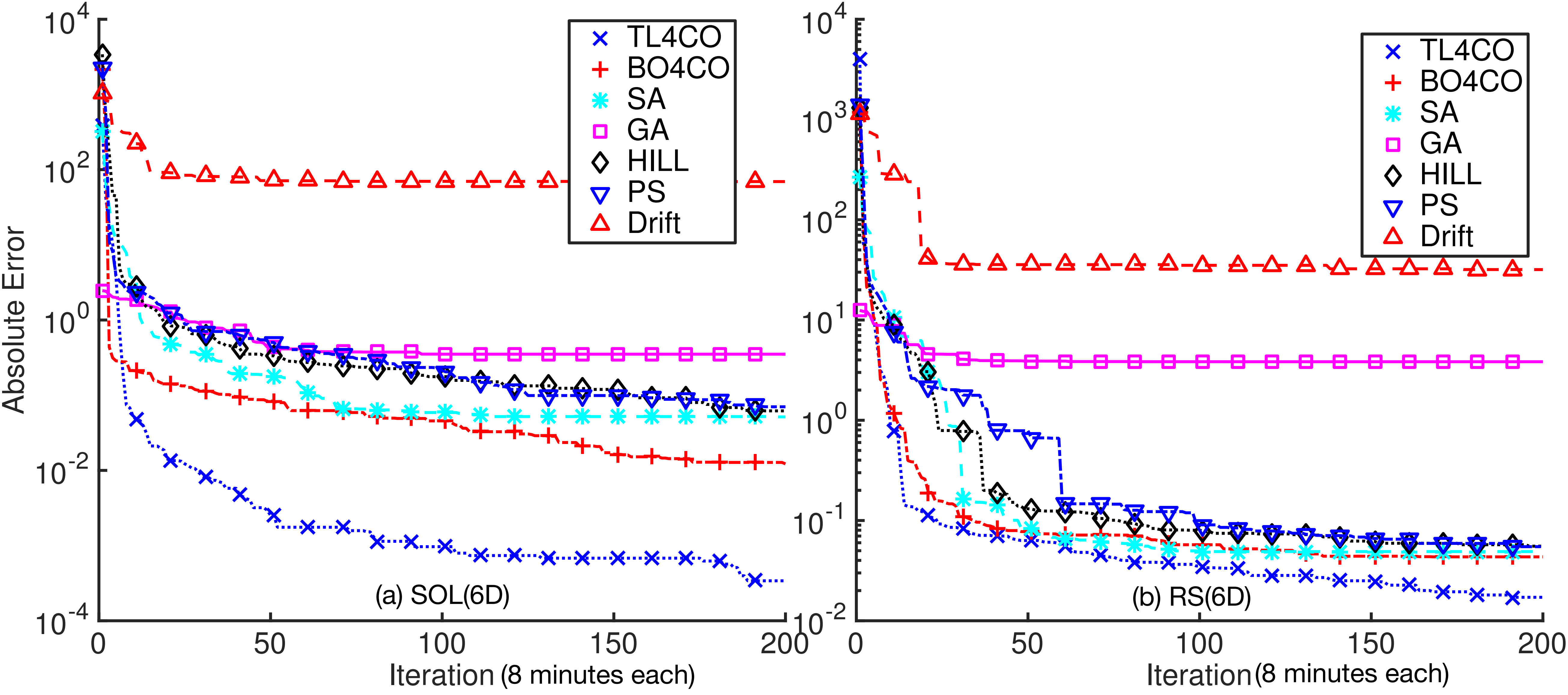
- 30 runs, report average performance
- Yes, we did full factorial measurements and we know where global min is... ☺



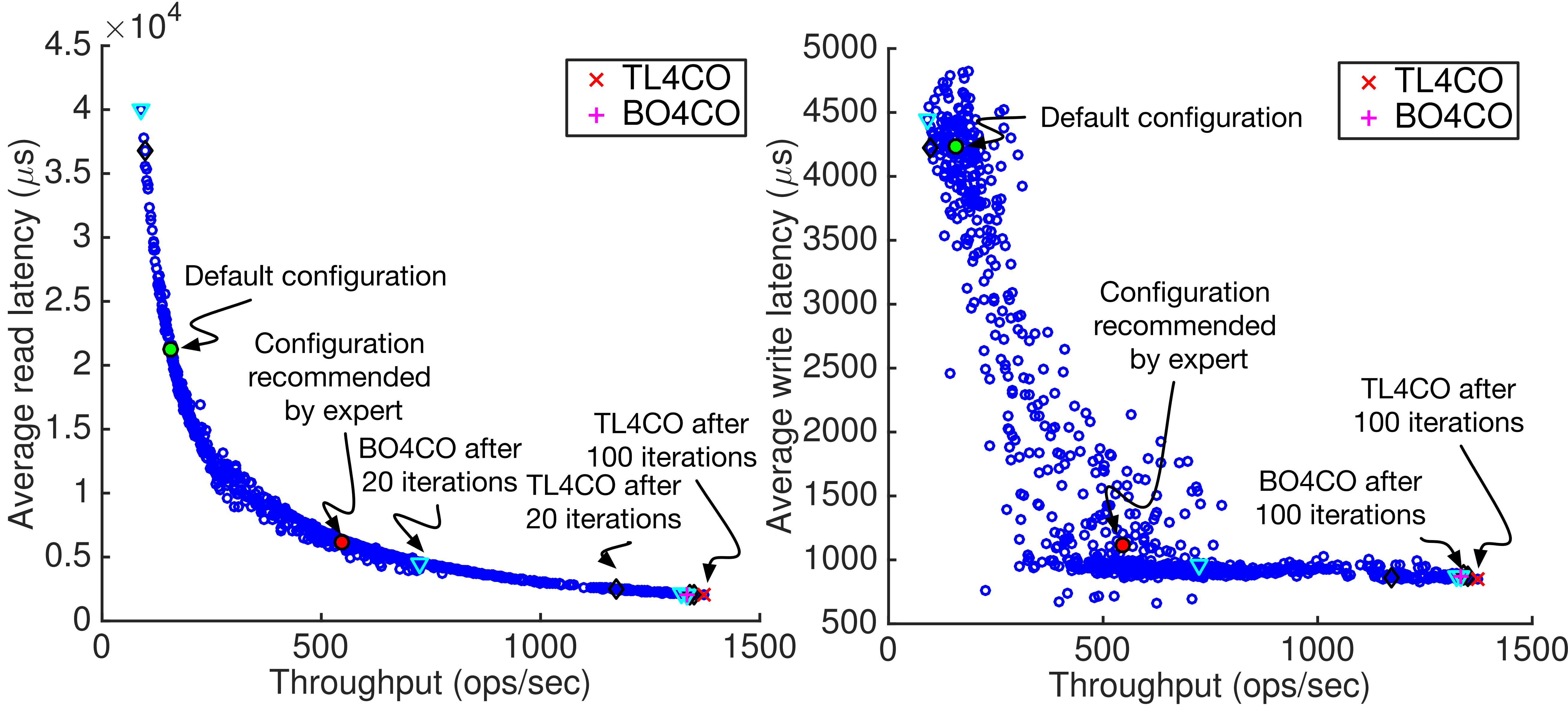
Experimental results (cont.)



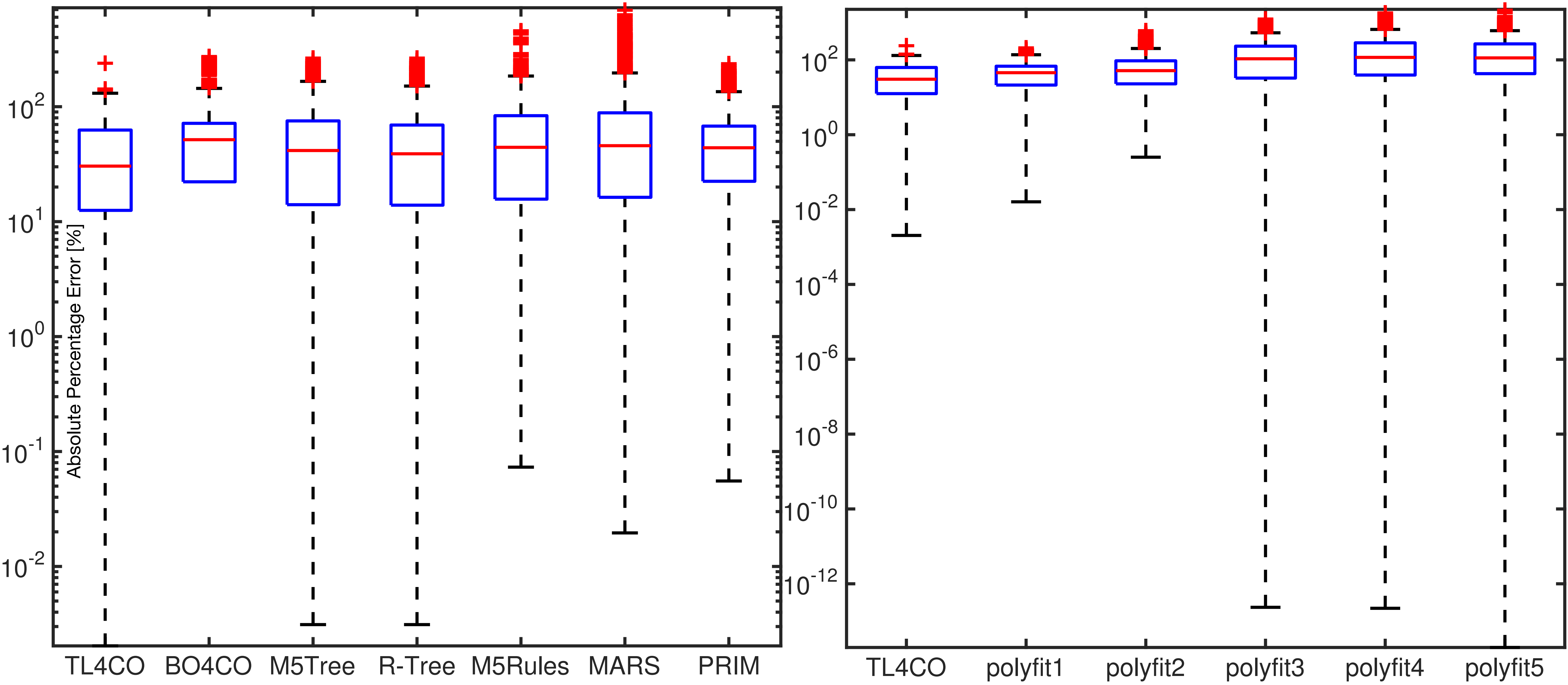
Experimental results (cont.)



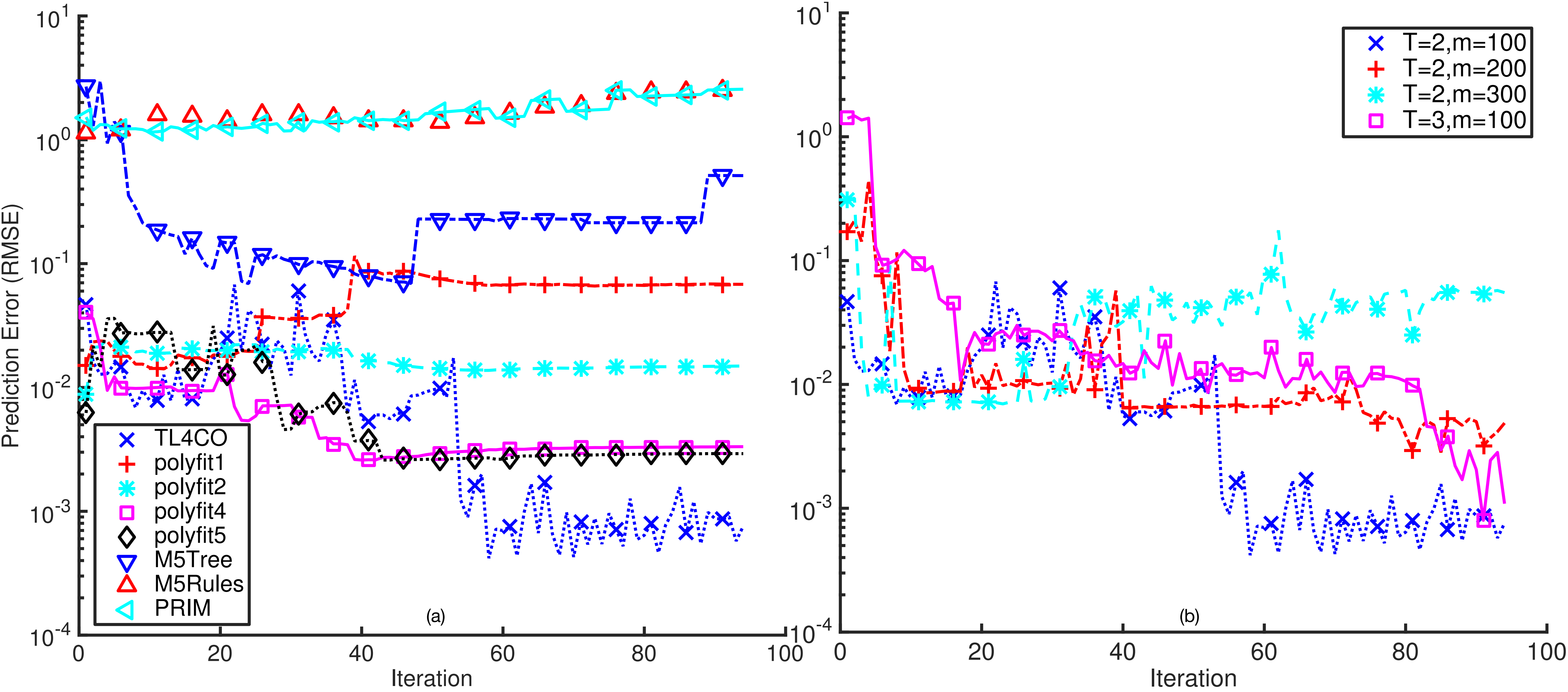
Comparison with default and expert prescription



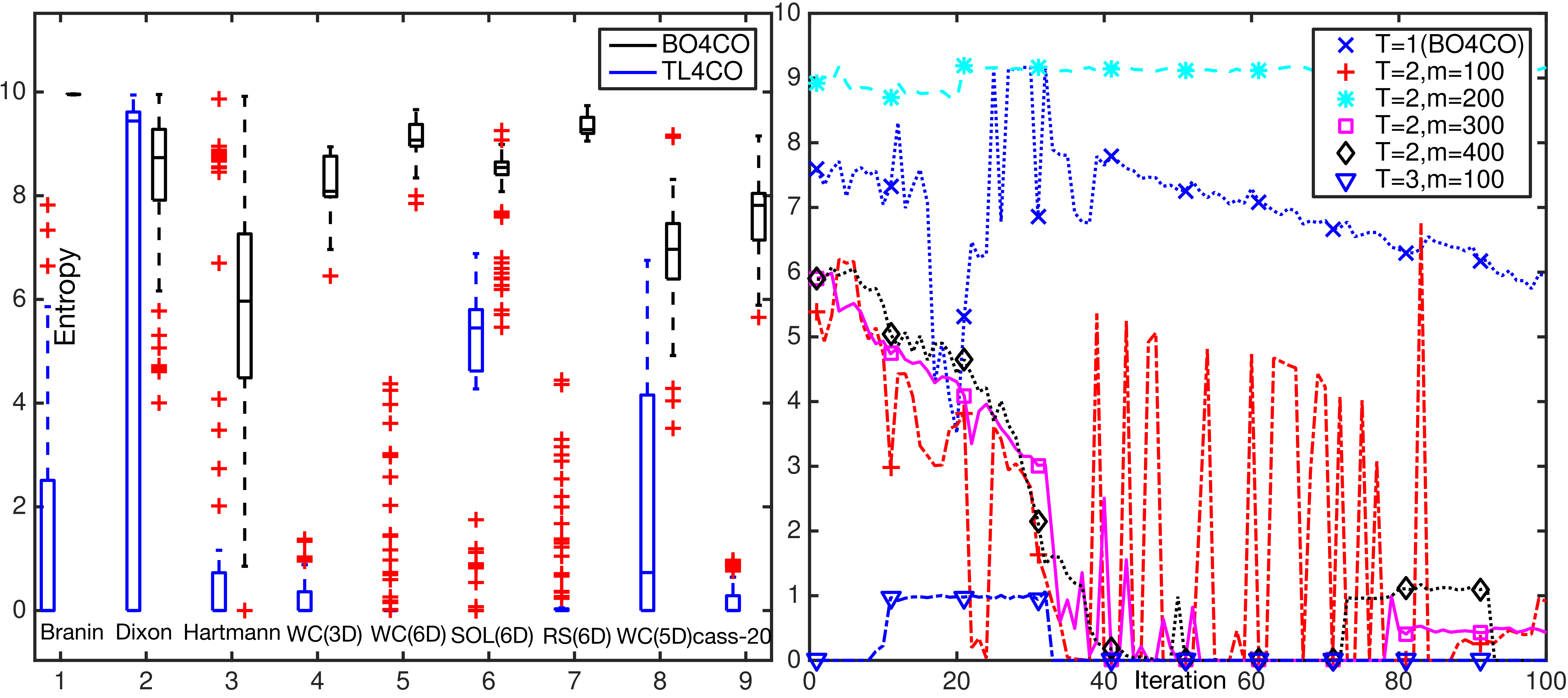
Model accuracy



Prediction accuracy over time

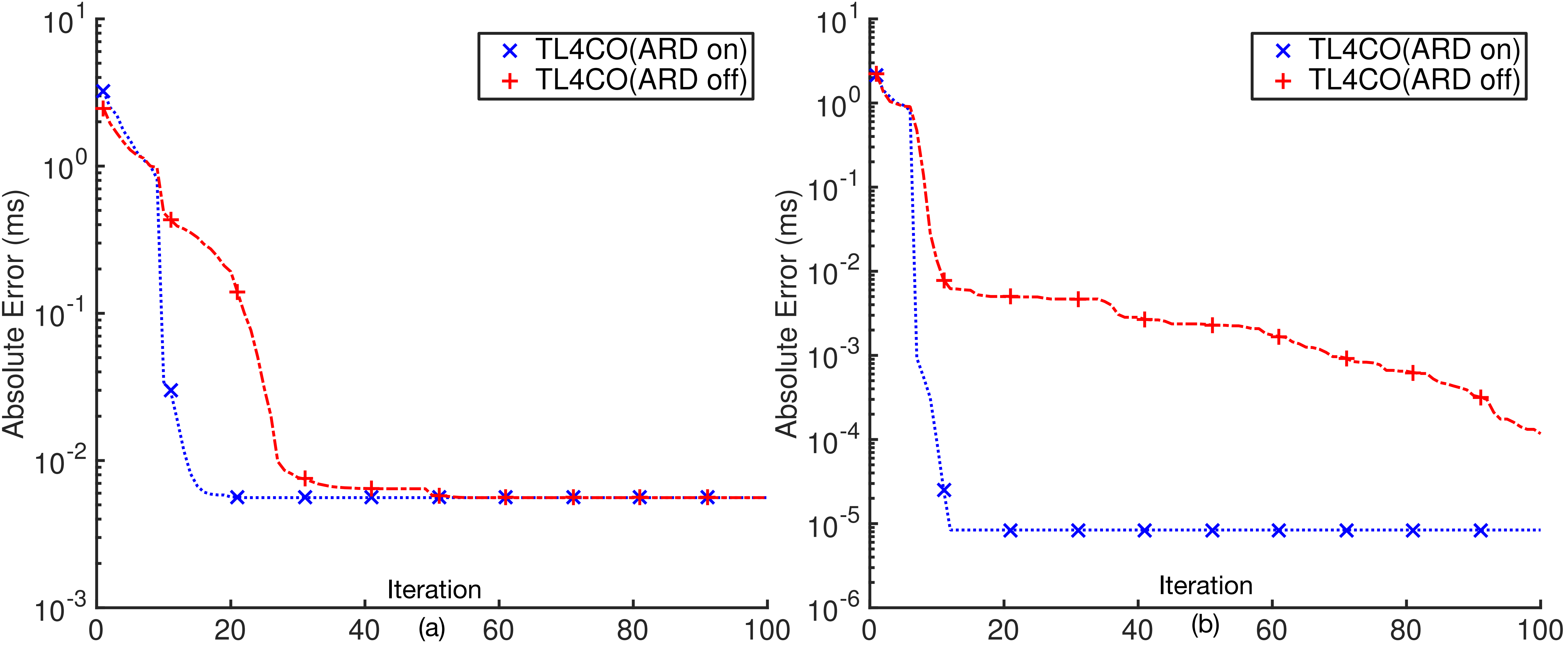


Entropy of the density function of the minimizers

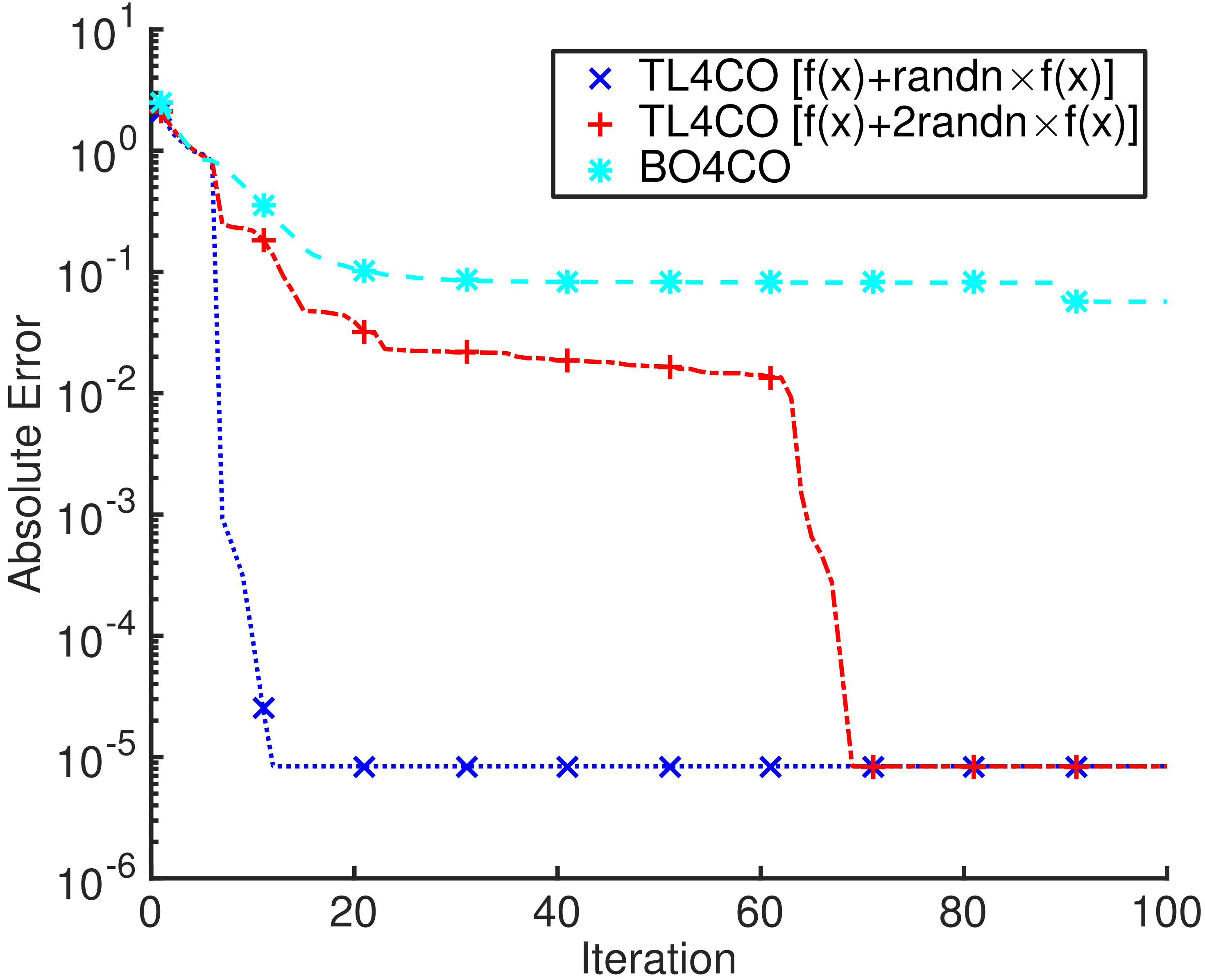


Effect of ARD

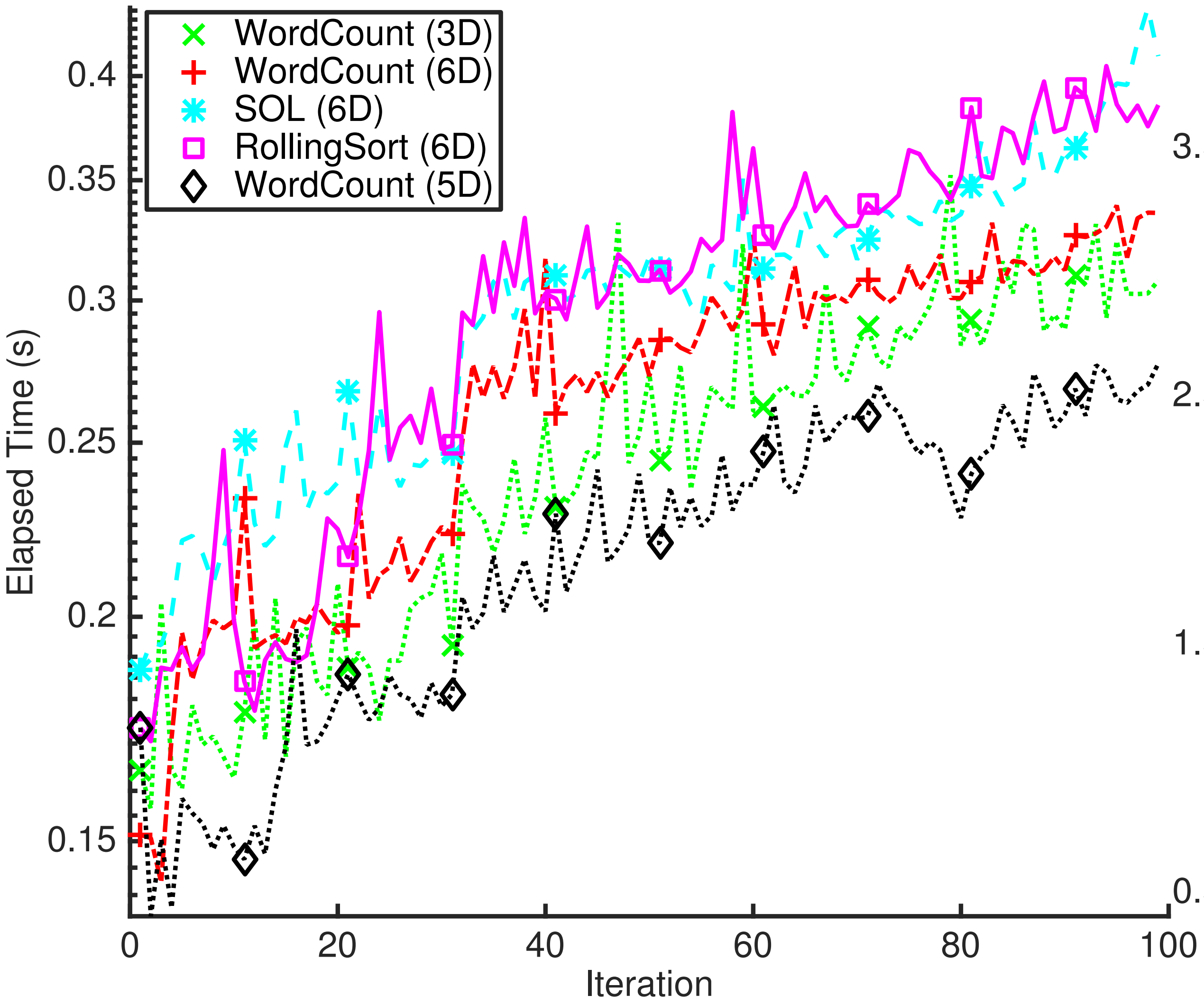
$$k_{xx}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\Sigma_{\ell=1}^{d}(-\theta_\ell \delta(\boldsymbol{x}_i \neq \boldsymbol{x}_j))),$$



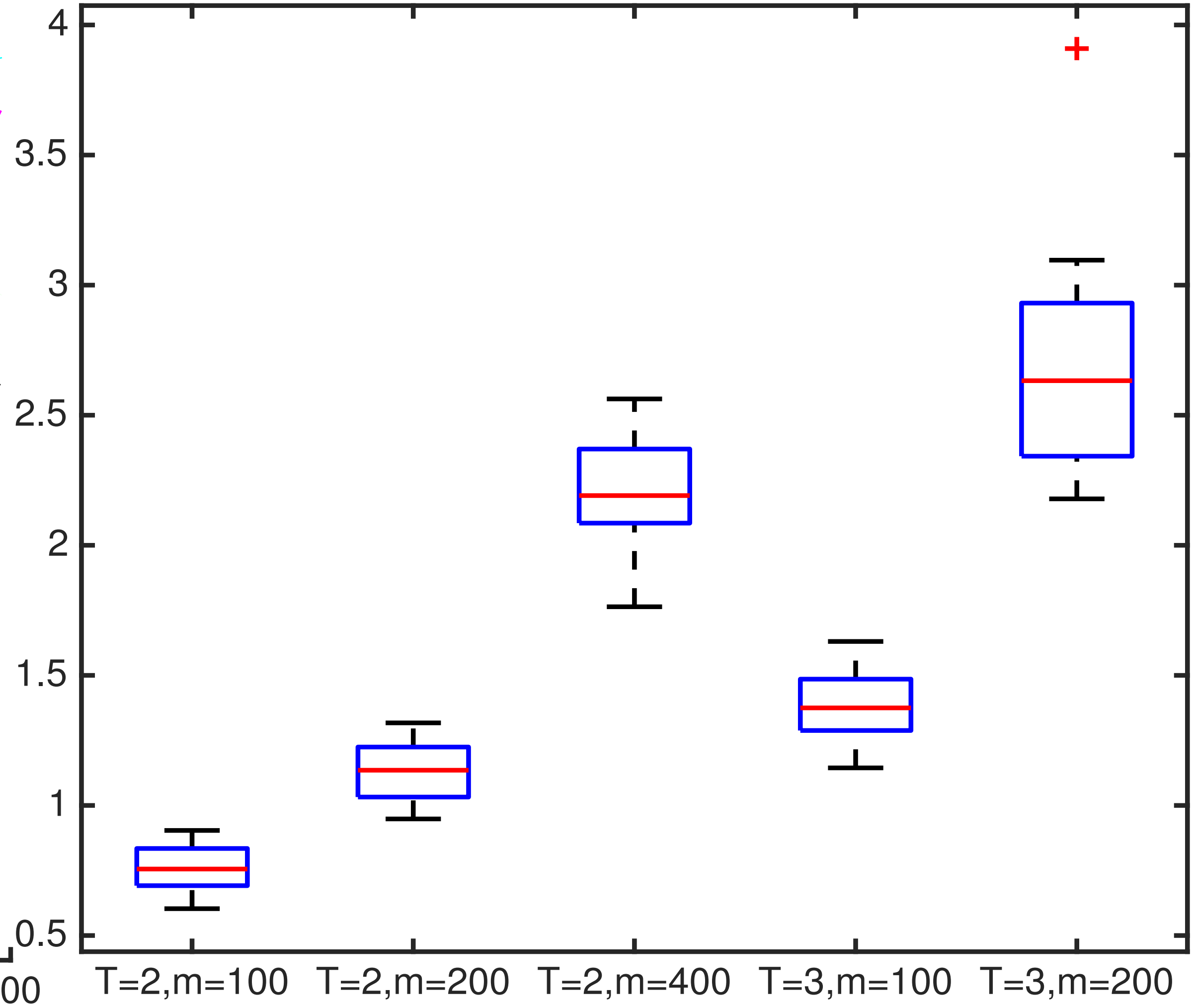
Effect of correlation between tasks



Runtime overhead



(a) TL4CO runtime for different datasets

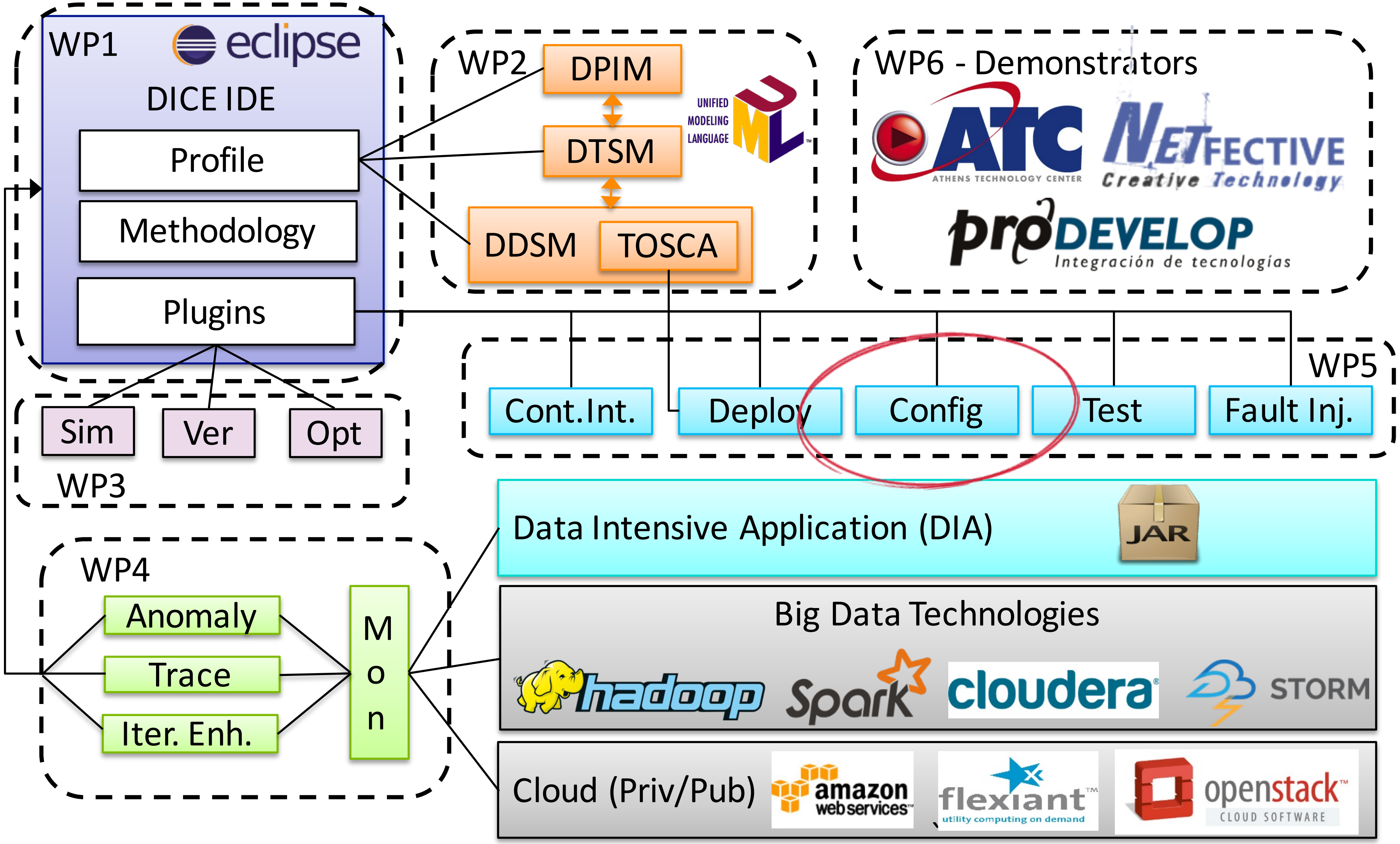


(b) TL4CO runtime for different size of observations

Key takeaways

- A principled way to leverage prior knowledge gained from searches over previous versions of the system.
- Multi-task GPs can be used in order to capture correlation between related tuning tasks.
- MTGPs are more accurate than STGP and other regression models.
- Application to SPSs, Batch and NoSQL
- Lead to a better performance in practice

Acknowledgement: BO4CO and TL4CO are now integrated with other DevOps tools in the delivery pipeline for Big Data in H2020 DICE project (http://www.dice-h2020.eu/)



Tool Demo: http://www.slideshare.net/pooyanjamshidi/configuration-optimization-tool