

How much data is sufficient to learn high-performing algorithms?

Ellen Vitercik
Carnegie Mellon University

STOC'21



Nina Balcan



Dan DeBlasio



Travis Dick



Carl Kingsford



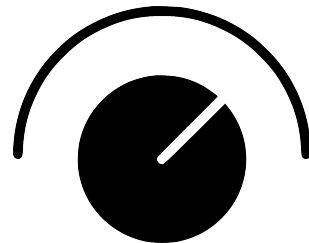
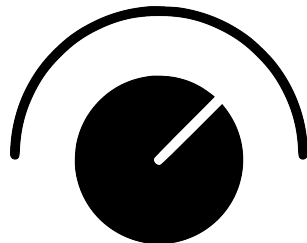
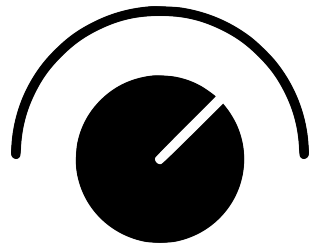
Tuomas Sandholm

Data-driven algorithm design

Algorithms often have **many tunable parameters**

Significant impact on runtime, solution quality, ...

Hand-tuning is **time-consuming**, **tedious**, and **error prone**

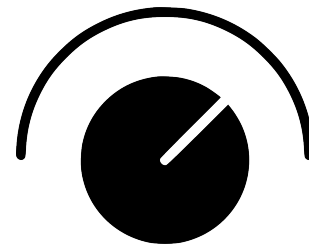
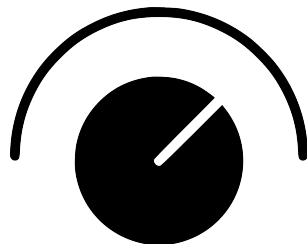
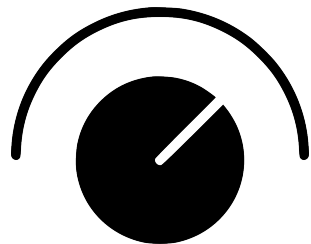


Data-driven algorithm design

Goal: Automate algorithm configuration via machine learning
Algorithmically find good parameter settings
using a set of "typical" inputs from application at hand

Training set

Parameter setting should—**ideally**—be good on future inputs



Example: Sequence alignment

Goal: Line up pairs of strings

Applications: Biology, natural language processing, etc.



Did you mean: [vitercik](#)

Sequence alignment algorithms

Input: Two sequences S and S'

Output: Alignment of S and S'

$S = A \ C \ T \ G$
 $S' = G \ T \ C \ A$

Gap
↓
A - - C T G
- G T C A -
↑ ↑ ↑
Insertion/deletion (*indel*) Match Mismatch

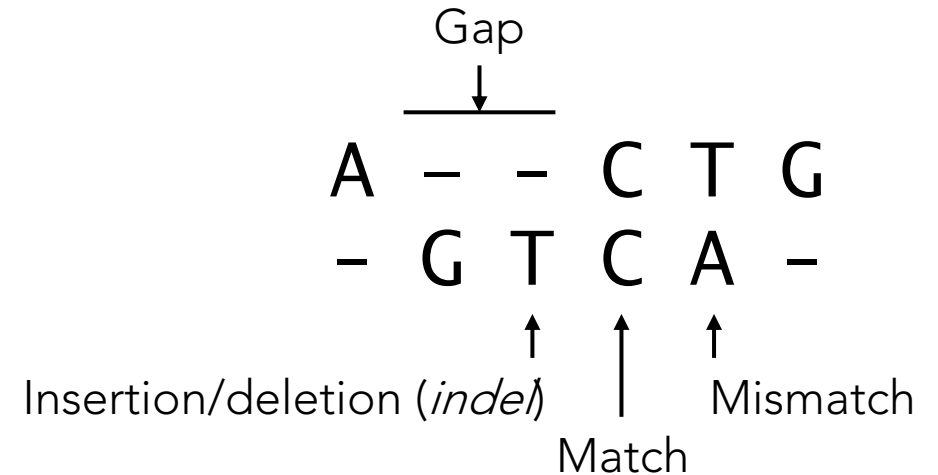
Sequence alignment algorithms

Standard algorithm with parameters $\rho_1, \rho_2, \rho_3 \geq 0$:

Return alignment maximizing:

$$(\# \text{ matches}) - \rho_1 \cdot (\# \text{ mismatches}) - \rho_2 \cdot (\# \text{ indels}) - \rho_3 \cdot (\# \text{ gaps})$$

$S = A \ C \ T \ G$
 $S' = G \ T \ C \ A$



Sequence alignment algorithms


Can sometimes access **ground-truth, reference** alignment

E.g., in computational biology: Bahr et al., Nucleic Acids Res.'01; Raghava et al., BMC Bioinformatics '03; Edgar, Nucleic Acids Res.'04; Walle et al., Bioinformatics'04

Requires extensive manual alignments
...rather just run parameterized algorithm

How to tune algorithm's parameters?

*"There is **considerable disagreement**
among molecular biologists about the
correct choice"* [Gusfield et al. '94]



A	-	-	C	T	G
-	G	T	C	A	-

Sequence alignment algorithms

-GRTCPKPDDL PFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYPAPKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLG NWSAMPSC-KA

Ground-truth alignment of protein sequences

Sequence alignment algorithms

-GRTCPKPDDL PFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYPAPKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLGNEWSAMPSC-KA

Ground-truth alignment of protein sequences

GRTCP---KPDDL PFSTVVPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDN-GFVNYPAPKPTLYYK-DKATFGCHDGY-SLDGPEEIECTKLGNEWS-AMPSCKA

Alignment by algorithm with **poorly-tuned** parameters

Sequence alignment algorithms

-GRTCPKPDDL PFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYP AKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLG NWSAMPSC-KA

Ground-truth alignment of protein sequences

GRTCP---KPDDL PFSTVVPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDN-GFVNYP AKPTLYYK-DKATFGCHDGY-SLDGPEEIECTKLG NWS-AMPSCKA

Alignment by algorithm with **poorly-tuned** parameters

GRTCPKPDDL PFSTV-VPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDNGFVNYP AKPTLYYKDKATFGCHDGY-SLDGPEEIECTKLG NWSA-MPSCKA

Alignment by algorithm with **well-tuned** parameters

Automated configuration

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs



3. Find parameters with good performance on average over T
Runtime, solution quality, etc.

Automated configuration

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs



3. Find parameters with good performance on average over T

Output alignment is close to reference alignment

Automated configuration

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs

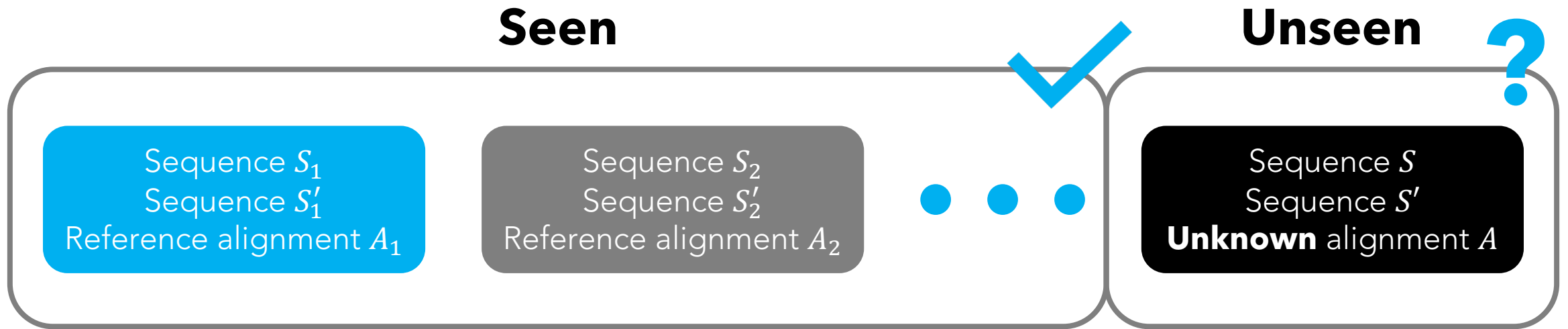


3. Find parameters with good performance on average over T

Key question (focus of talk):

Will those parameters have good **future** performance?

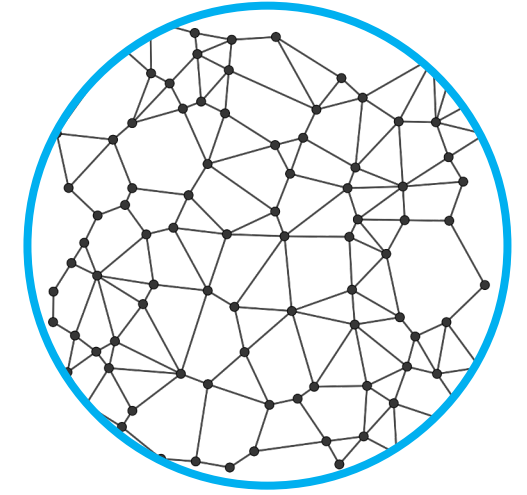
Automated configuration



Key question (focus of talk):

Will those parameters have good **future** performance?

Automated configuration



Model applies in **many** settings, including:

Constraint satisfaction problems, e.g.:

Horvitz, Ruan, Gomes, Krautz, Selman, Chickering

Nudelman, Leyton-Brown, Hoos, Devkar, Shoham

Sayag, Fine, Mansour

Hutter, Hamadi, Hoos, Leyton-Brown

Xu, Hutter, Leyton-Brown

Xu, Hutter, Hoos, Leyton-Brown

Xu, Hoos, Leyton-Brown

Kleinberg, Leyton-Brown, Lucier

Balcan, Dick, Sandholm, **Vitercik**

Weisz, György, Szepesvári

Kleinberg, Leyton-Brown, Lucier, Graham

UAI'01

CP'04

STACS'06

CP'06

CP'07

JAIR'08

AAAI'10

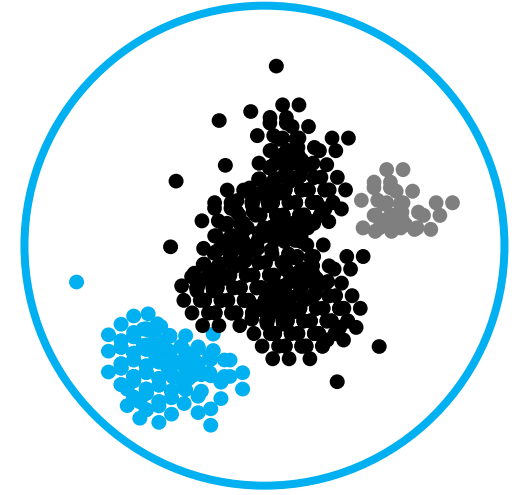
IJCAI'17

ICML'18

ICML '18, '19

NeurIPS'19

Automated configuration



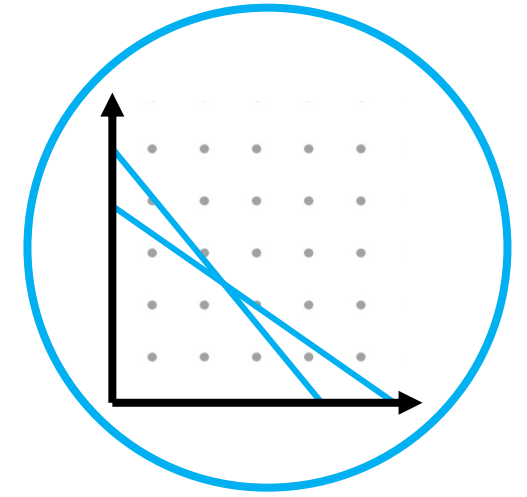
Model applies in **many** settings, including:

Clustering, e.g.:

Balcan, Nagarajan, **Vitercik**, White
Balcan, Dick, White
Garg, Kalai
Balcan, Dick, Lang

COLT'17
NeurIPS'18
NeurIPS'18
ICLR'20

Automated configuration

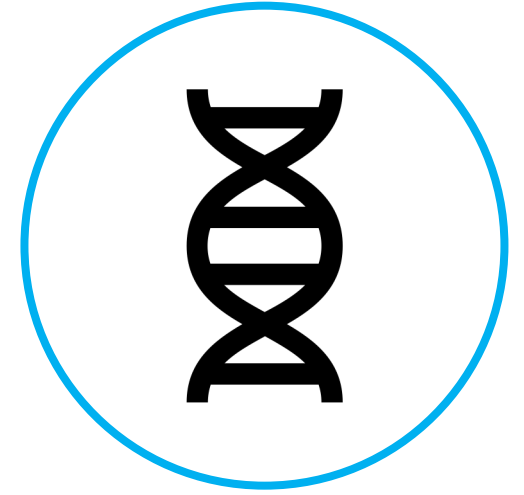


Model applies in **many** settings, including:

Integer and linear programming, e.g.:

Leyton-Brown, Nudelman, Andrew, McFadden, Shoham	IJCAI'03, CP'03
Hutter, Hoos, Leyton-Brown, Stützle	JAIR'09
Hutter, Hoos, Leyton-Brown	LION'11, AIJ'14
Sandholm	Handbook of Market Design'13
He, Daume, Eisner	NeurIPS'14
Khalil, Le Bodic, Song, Nemhauser, Dilkina	AAAI'16
Balcan, Nagarajan, Vitercik , White	COLT'17
Balcan, Dick, Sandholm, Vitercik	ICML'18
Balcan, Dick, Vitercik	FOCS'18
Balcan, Sandholm, Vitercik	AAAI'20
Balcan, Sandholm, Vitercik	ICML'20

Automated configuration



Model applies in **many** settings, including:

Computational biology, e.g.:

Majoros, Salzberg

Chikhi, Medvedev

May, Tamura, Noble

DeBlasio, Kececioglu

DeBlasio, Kim, Kingsford

Balcan, DeBlasio, Dick, Kingsford, Sandholm, **Vitercik**

Bioinformatics'04

Bioinformatics'13

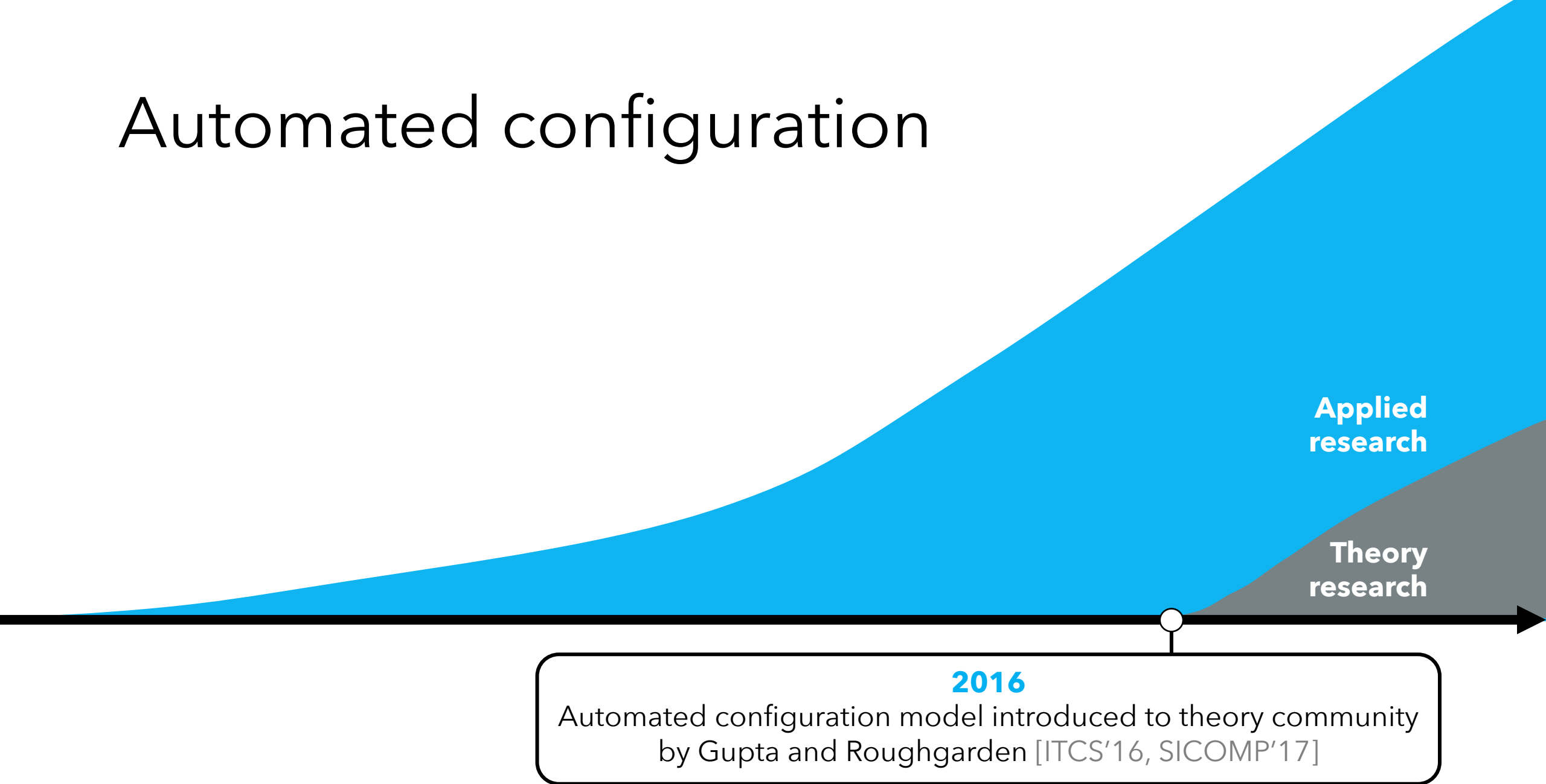
J. of Proteome Research'17

Springer'18

WBC@ICML'19

'20

Automated configuration



This talk: Main result

Key question (focus of talk):

Good performance on **average** over **training set** implies good **future** performance?

Answer this question for any parameterized algorithm where:

Performance is **piecewise-structured** function of parameters

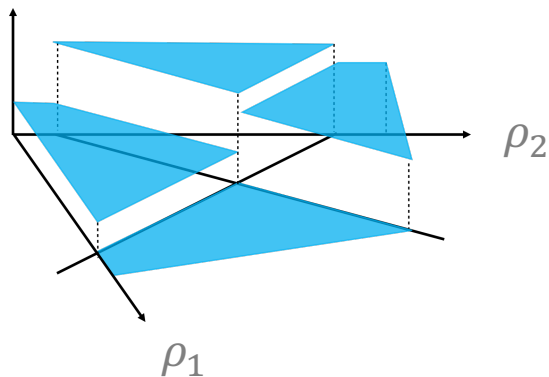
Piecewise constant, linear, quadratic, ...

This talk: Main result

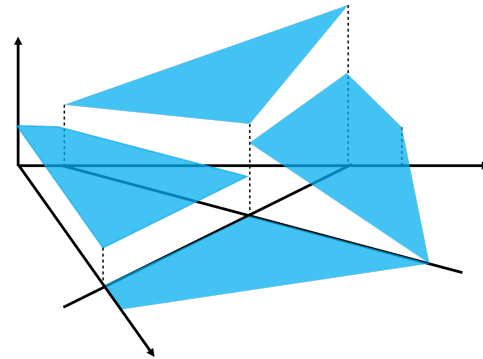
Performance is piecewise-structured function of parameters

Piecewise constant, linear, quadratic, ...

Algorithmic
performance
on fixed input



Piecewise constant



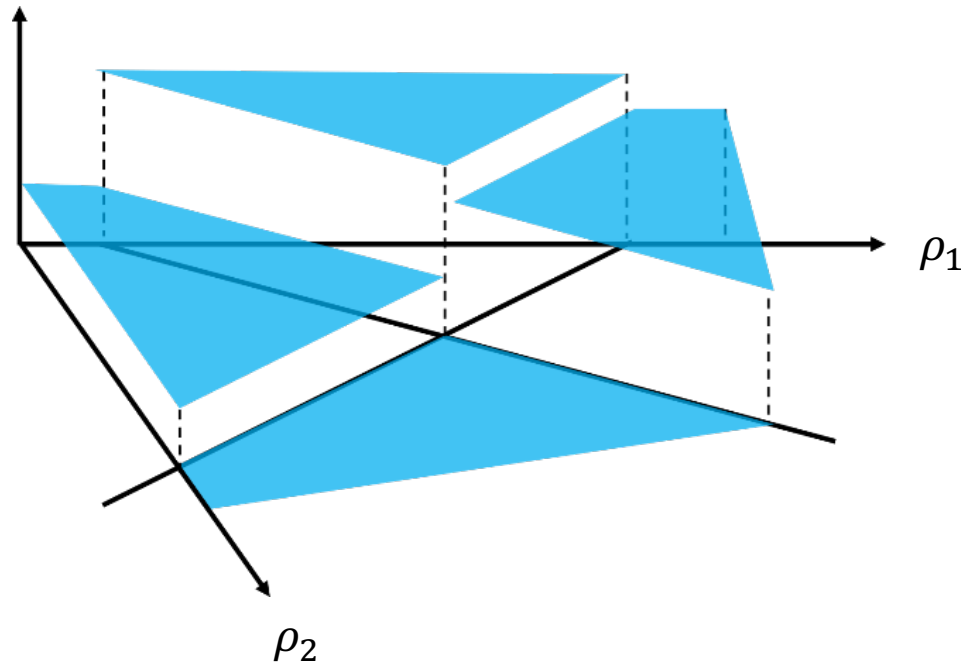
Piecewise linear



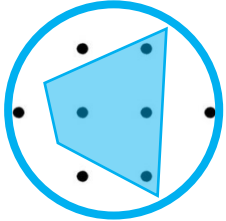
Piecewise ...

Example: Sequence alignment

Distance between **algorithm's output** given S, S'
and **ground-truth** alignment is p-wise constant

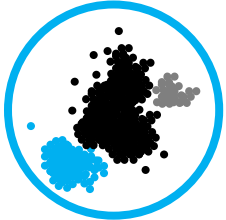


Domains with piecewise structure



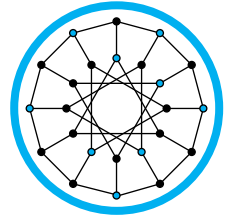
Integer programming

Balcan, Dick, Sandholm, **Vitercik**, ICML'18;
Balcan, Nagarajan, **Vitercik**, White, COLT'17



Clustering

Balcan, Nagarajan, **Vitercik**, White, COLT'17
Balcan, Dick, White, NeurIPS'18; Balcan, Dick, Lang, ICLR'20



Greedy algorithms

Gupta, Roughgarden, ITCS'16



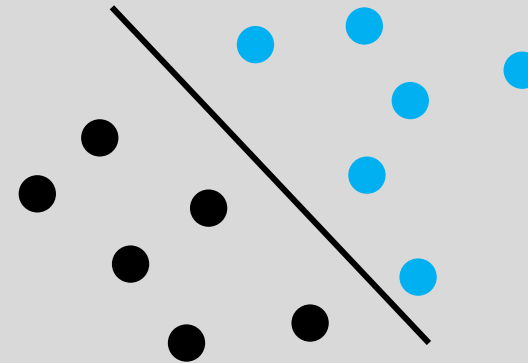
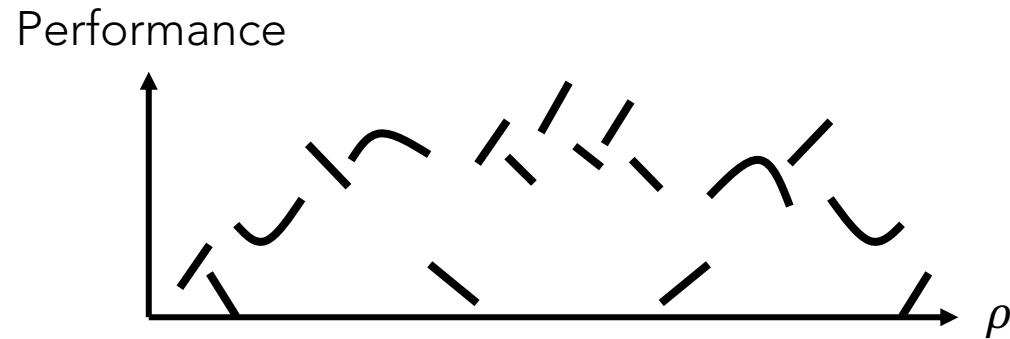
Computational biology

Balcan, DeBlasio, Dick, Kingsford, Sandholm, **Vitercik**, STOC'21

Primary challenge in combinatorial domains:

Algorithmic performance is a **volatile** function of parameters

Complex connection between parameters and performance



For well-understood functions in machine learning theory:

Simple connection between function parameters and value

Outline

1. Introduction
- 2. Model and problem formulation**
3. Our guarantees
4. Conclusions

Model

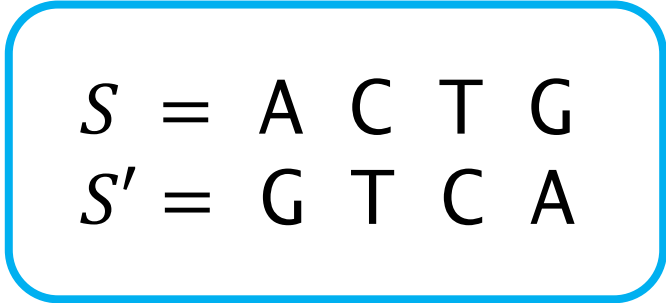
\mathbb{R}^d : Set of all parameters

\mathcal{X} : Set of all inputs

Example: Sequence alignment

\mathbb{R}^3 : Set of alignment algorithm parameters

\mathcal{X} : Set of sequence pairs


$$\begin{array}{l} S = A \ C \ T \ G \\ S' = G \ T \ C \ A \end{array}$$

One sequence pair $x = (S, S') \in \mathcal{X}$

Algorithmic performance

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
E.g., runtime, solution quality, distance to ground truth, ...

Assume $u_{\boldsymbol{\rho}}(x) \in [-1, 1]$

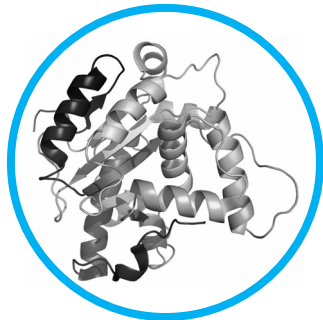
Can be generalized to $u_{\boldsymbol{\rho}}(x) \in [-H, H]$

Model

Standard assumption: Unknown distribution \mathcal{D} over inputs
Distribution models specific application domain at hand



E.g., distribution over pairs of DNA strands



E.g., distribution over pairs of protein sequences

Generalization bounds

Key question: For any parameter setting ρ ,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any ρ ,

Generalization bounds

Key question: For any parameter setting ρ ,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any ρ ,

$$\left| \underbrace{\frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i)}_{\text{Empirical average utility}} - \mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)] \right| \leq ?$$

Generalization bounds

Key question: For any parameter setting ρ ,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any ρ ,

$$\left| \frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i) - \underbrace{\mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)]}_{\text{Expected utility}} \right| \leq ?$$

Generalization bounds

Key question: For any parameter setting ρ ,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any ρ ,

$$\left| \frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)] \right| \leq ?$$

Strong **average** utility  Strong **expected** utility

Generalization bounds

Key question: For any parameter setting ρ ,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any ρ ,

$$\left| \frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)] \right| \leq ?$$

Strong **average** utility \rightarrow Strong **future** utility



Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
 - a. Example of piecewise-structured utility function**
 - b. Piecewise-structured functions more formally
 - c. Main theorem
 - d. Application: Sequence alignment
4. Conclusions

Sequence alignment algorithms

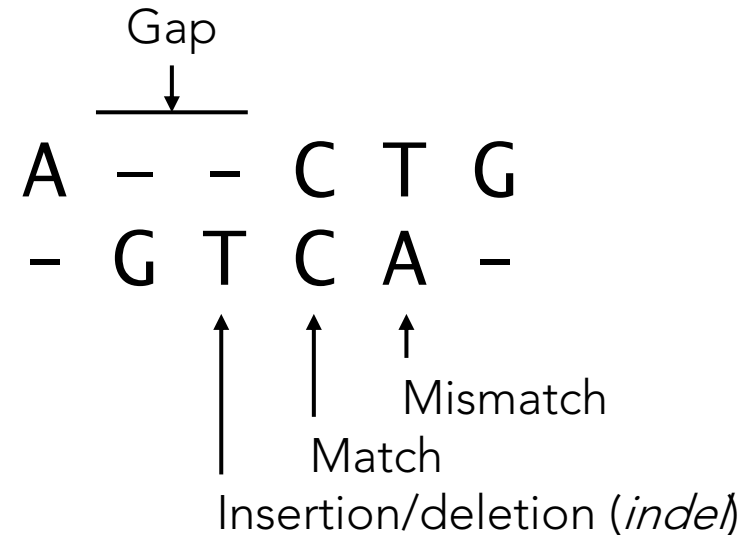
Standard algorithm with parameters $\rho_1, \rho_2, \rho_3 \geq 0$:

Return alignment maximizing:

$$(\# \text{ matches}) - \rho_1 \cdot (\# \text{ mismatches}) - \rho_2 \cdot (\# \text{ indels}) - \rho_3 \cdot (\# \text{ gaps})$$

$S = A \ C \ T \ G$

$S' = G \ T \ C \ A$

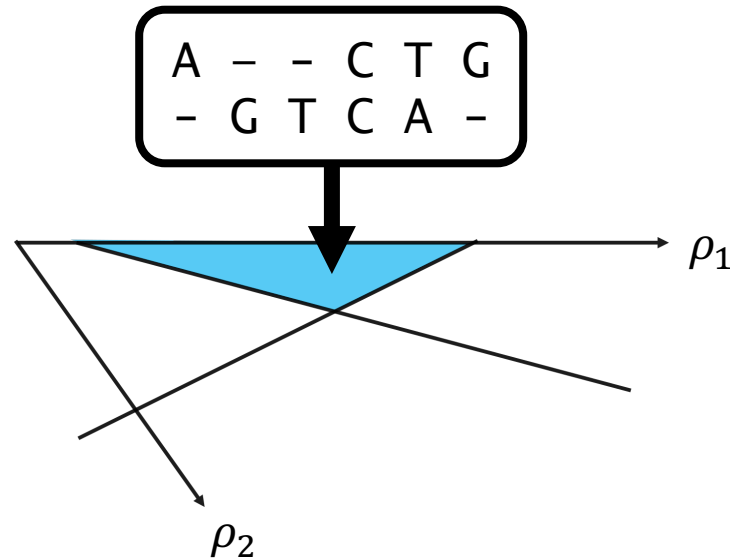


Sequence alignment algorithms

Lemma:

For any pair S, S' , there's a small partition of \mathbb{R}^3 s.t. in any region, algorithm's output is fixed across all parameters in region

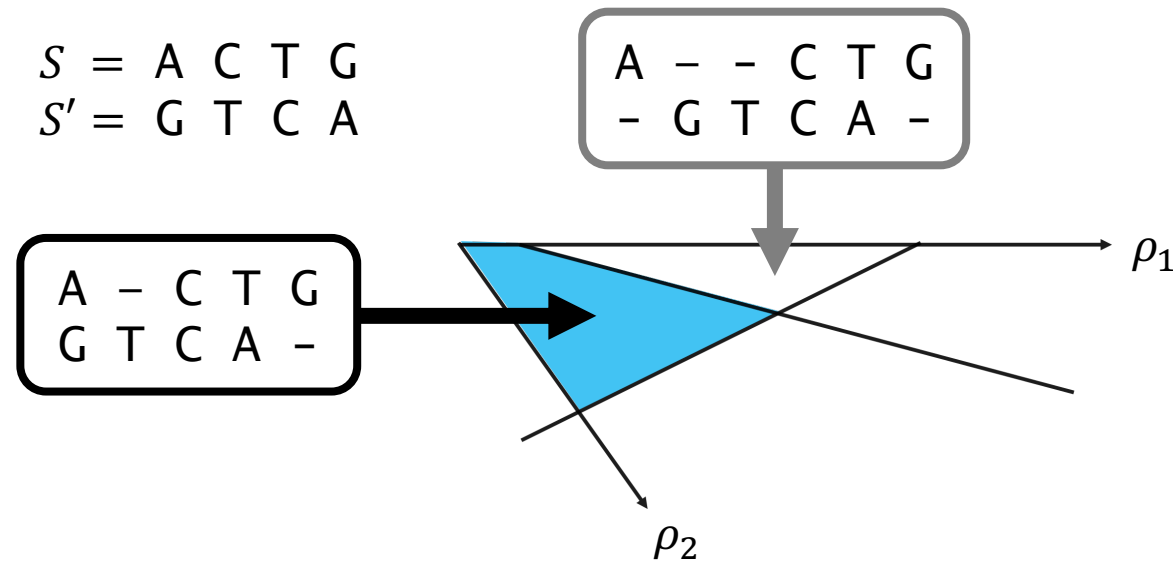
$S = A \ C \ T \ G$
 $S' = G \ T \ C \ A$



Sequence alignment algorithms

Lemma:

For any pair S, S' , there's a small partition of \mathbb{R}^3 s.t. in any region, algorithm's output is fixed across all parameters in region

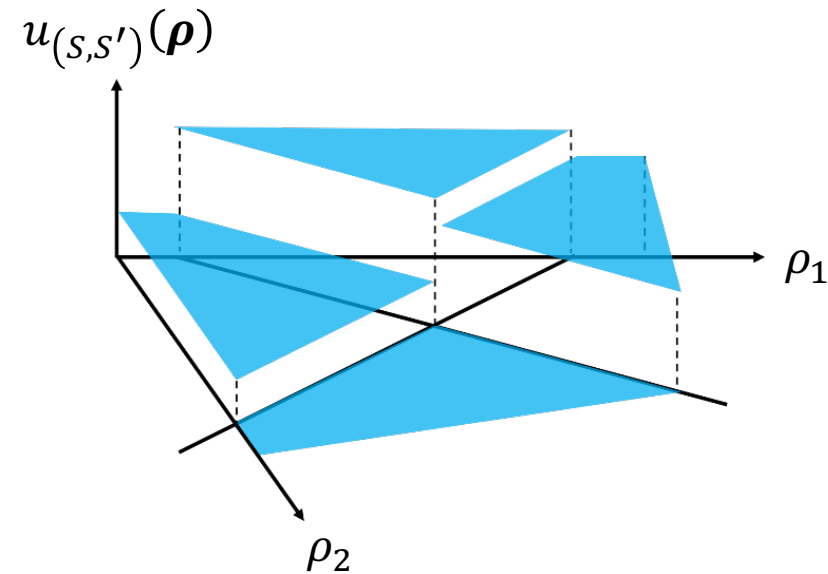


Piecewise-constant utility function

Corollary:

Utility is piecewise constant function of parameters

Distance between algorithm's output and ground-truth alignment



Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
 - a. Example of piecewise-structured utility function
 - b. Piecewise-structured functions more formally**
 - c. Main theorem
 - d. Application: Sequence alignment
4. Conclusions

Primal & dual classes

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
 $\mathcal{U} = \{u_{\boldsymbol{\rho}}: \mathcal{X} \rightarrow \mathbb{R} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$ **“Primal” function class**

Typically, prove guarantees by bounding complexity of \mathcal{U}

VC dimension, pseudo-dimension, Rademacher complexity, ...

Primal & dual classes

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
 $\mathcal{U} = \{u_{\boldsymbol{\rho}}: \mathcal{X} \rightarrow \mathbb{R} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$ **"Primal" function class**

Typically, prove guarantees by bounding **complexity** of \mathcal{U}

Challenge: \mathcal{U} is gnarly

E.g., in sequence alignment:

- Each domain element is a pair of sequences
- Unclear how to plot or visualize functions $u_{\boldsymbol{\rho}}$
- No obvious notions of Lipschitz continuity or smoothness to rely on

Primal & dual classes

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
 $\mathcal{U} = \{u_{\boldsymbol{\rho}}: \mathcal{X} \rightarrow \mathbb{R} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$ **“Primal” function class**

$u_x^*(\boldsymbol{\rho})$ = utility as function of parameters

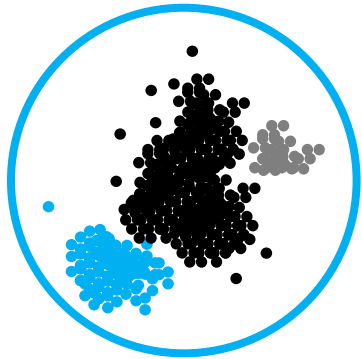
$$u_x^*(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$$

$\mathcal{U}^* = \{u_x^*: \mathbb{R}^d \rightarrow \mathbb{R} \mid x \in \mathcal{X}\}$ **“Dual” function class**

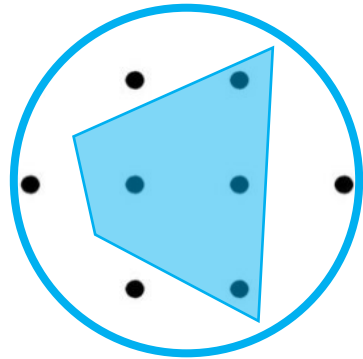
- Dual functions have simple, Euclidean domain
- Often have ample structure can use to bound complexity of \mathcal{U}

Piecewise-structured functions

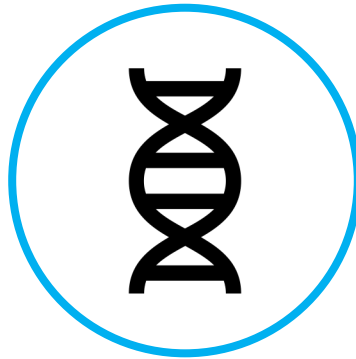
Dual functions $u_x^*: \mathbb{R}^d \rightarrow \mathbb{R}$ are **piecewise-structured**



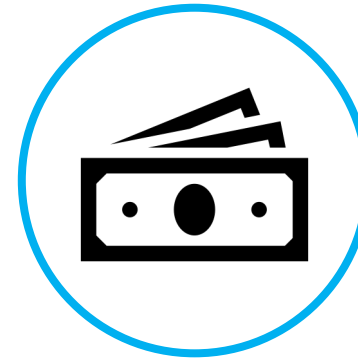
Clustering
algorithm
configuration



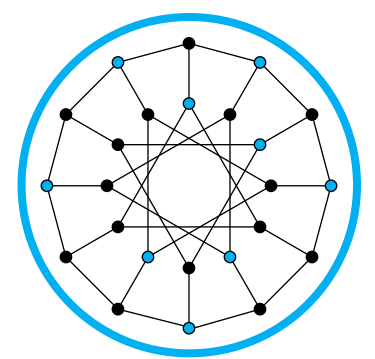
**Integer
programming**
algorithm
configuration



**Computational
biology**
algorithm
configuration



Mechanism
configuration



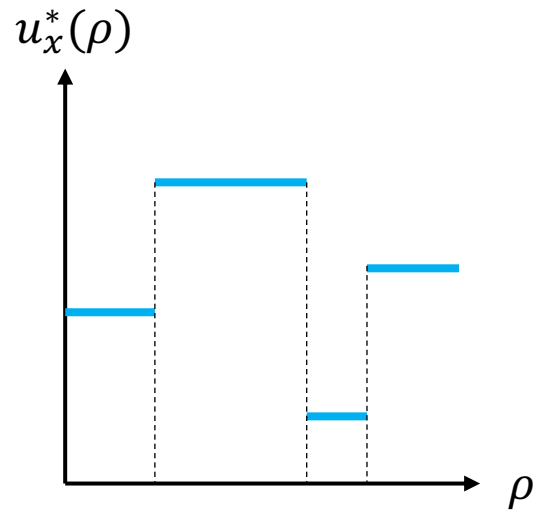
Greedy
algorithm
configuration

Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
 - a. Example of piecewise-structured utility function
 - b. Piecewise-structured functions more formally
 - c. Main theorem**
 - d. Application: Sequence alignment
4. Conclusions

Warmup: 1-dimensional parameters

For every input x , $u_x^*: \mathbb{R} \rightarrow \mathbb{R}$ is p-wise constant with $\leq k$ pieces.



Warmup: 1-dimensional parameters

For every input x , $u_x^*: \mathbb{R} \rightarrow \mathbb{R}$ is p-wise constant with $\leq k$ pieces.

Theorem:

Training set of size $\tilde{O}\left(\frac{\log k}{\epsilon^2}\right)$ implies that WHP, for all $\forall \rho$,
average utility over training set is ϵ -close to **expected** utility

$$\left| \frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)] \right| \leq \epsilon$$

Warmup: 1-dimensional parameters

For every input x , $u_x^*: \mathbb{R} \rightarrow \mathbb{R}$ is p-wise constant with $\leq k$ pieces.

Theorem:

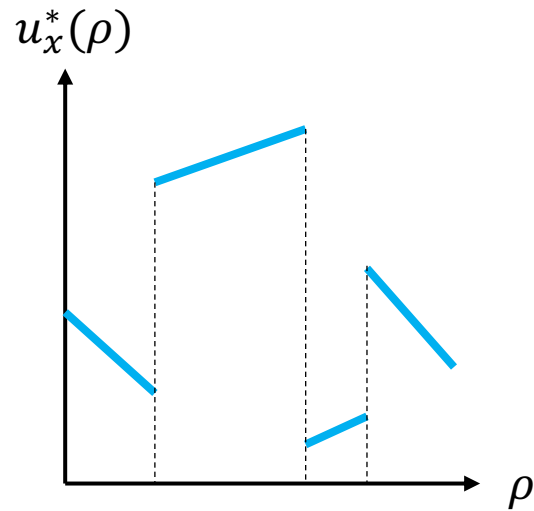
Training set of size $\tilde{O}\left(\frac{\log k}{\epsilon^2}\right)$ implies that WHP, for all $\forall \rho$,
average utility over training set is ϵ -close to **future** utility

$$\left| \frac{1}{N} \sum_{i=1}^N u_{\rho}(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[u_{\rho}(x)] \right| \leq \epsilon$$



Warmup: 1-dimensional parameters

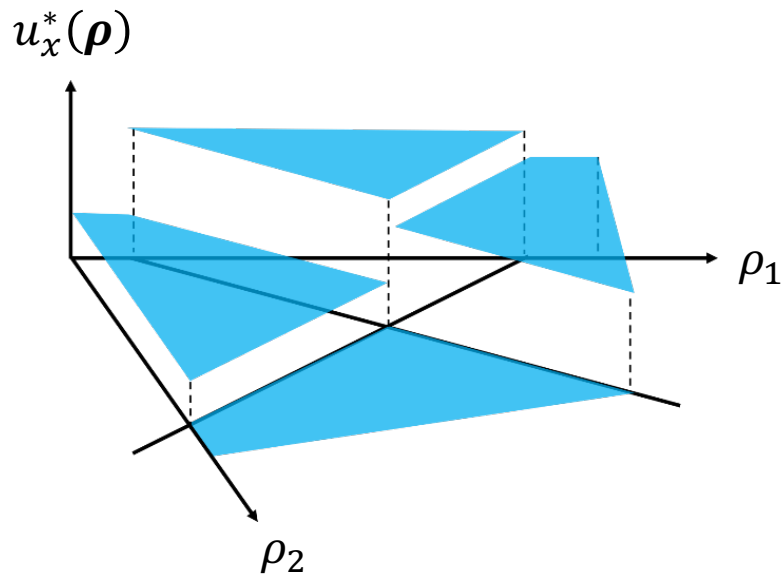
For every input x , $u_x^*: \mathbb{R} \rightarrow \mathbb{R}$ is ~~p-wise constant~~ with $\leq k$ pieces.
linear



Warmup: 1-dimensional parameters

For every input x , $u_x^*: \mathbb{R} \rightarrow \mathbb{R}$ is p-wise ~~constant~~ with $\leq k$ pieces.

???

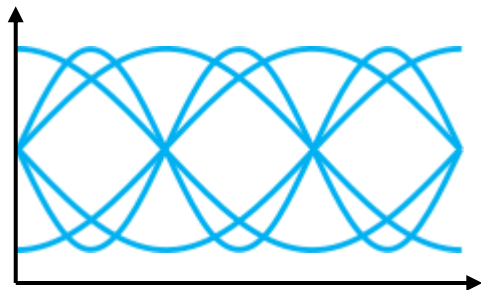


Multiple parameters?

Intrinsic complexity

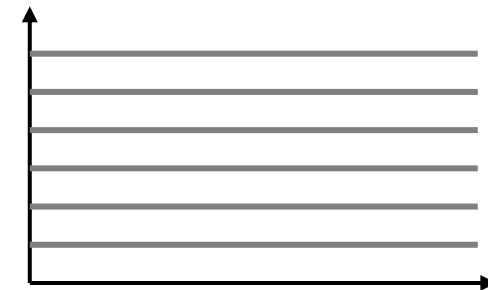
“Intrinsic complexity” $\mathcal{C}_{\mathcal{G}}$ of function class \mathcal{G}

- Measures how well functions in \mathcal{G} fit complex patterns
- Specific ways to quantify “intrinsic complexity”:
 - VC dimension
 - Pseudo-dimension



More complex

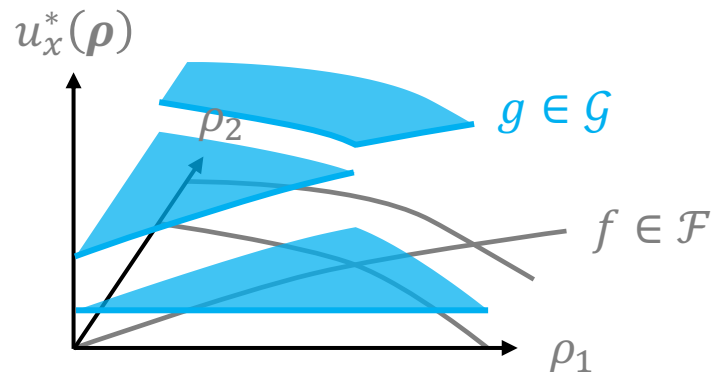
Less complex



Main result (informal)

Boundary functions $f_1, \dots, f_k \in \mathcal{F}$ partition \mathbb{R}^d s.t. in each region, $u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

Training set of size $\tilde{O}\left(\frac{1}{\epsilon^2}(C_{\mathcal{F}} + C_{\mathcal{G}})\log k\right)$ implies
WHP $\forall \boldsymbol{\rho}$, **avg** utility over training set is ϵ -close to **exp** utility



Main result (informal)

Boundary functions $f_1, \dots, f_k \in \mathcal{F}$ partition \mathbb{R}^d s.t. in each region, $u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

Training set of size $\tilde{O}\left(\frac{1}{\epsilon^2} (C_{\mathcal{F}} + C_{\mathcal{G}}) \log k\right)$ implies
WHP $\forall \boldsymbol{\rho}$, **avg** utility over training set is ϵ -close to **exp** utility

\mathcal{F}, \mathcal{G} are typically very well structured

- \mathcal{G} = set of all **constant** functions $\Rightarrow C_{\mathcal{G}} = O(1)$
- \mathcal{G} = set of all **linear** functions in \mathbb{R}^d $\Rightarrow C_{\mathcal{G}} = O(d)$

Main result (informal)

Boundary functions $f_1, \dots, f_k \in \mathcal{F}$ partition \mathbb{R}^d s.t. in each region, $u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

Theorem:

$$\text{Pdim}(\mathcal{U}) = \tilde{O}\left((\text{VCdim}(\mathcal{F}^*) + \text{Pdim}(\mathcal{G}^*)) \log k\right)$$

 **Primal** function class $\mathcal{U} = \{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$

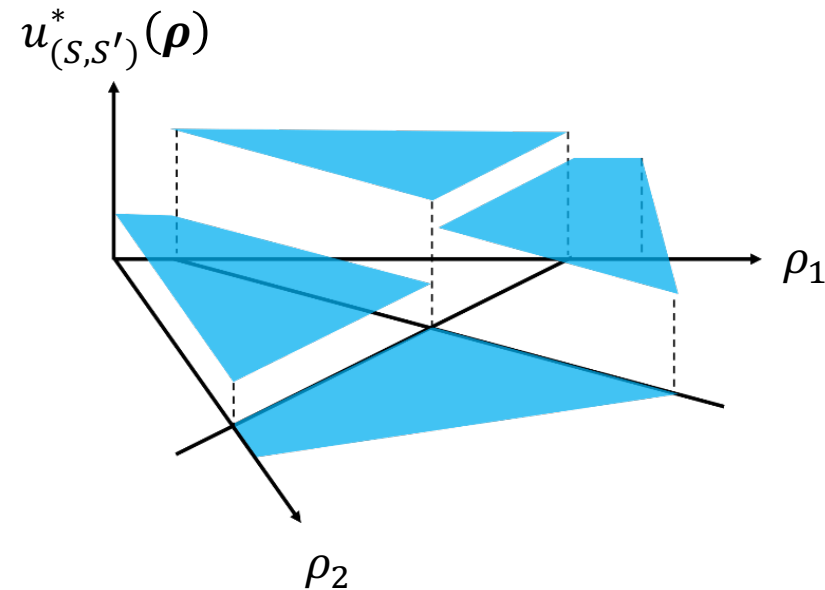
Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
 - a. Example of piecewise-structured utility function
 - b. Piecewise-structured functions more formally
 - c. Main theorem
 - d. Application: Sequence alignment**
4. Conclusions

Piecewise constant dual functions

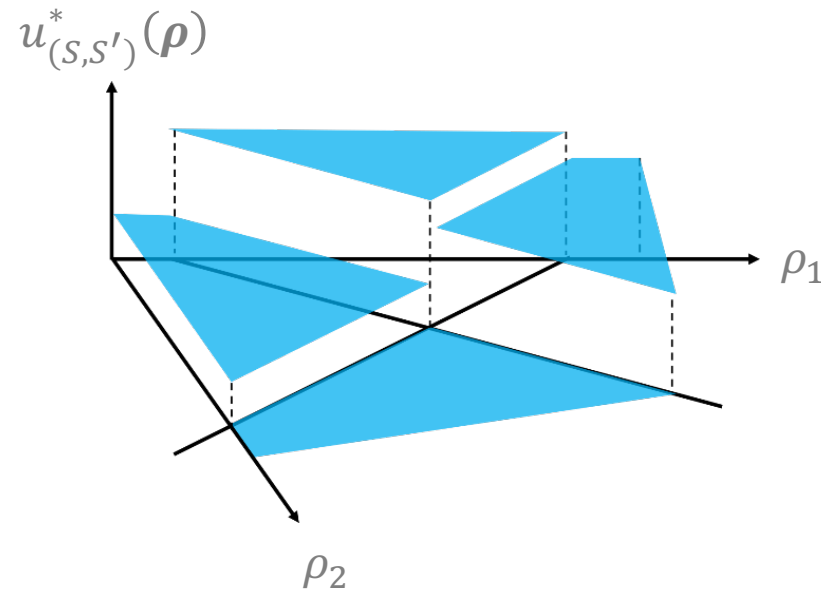
Lemma:

Utility is piecewise constant function of parameters



Sequence alignment guarantees

Theorem: Training set of size $\tilde{O}\left(\frac{\log(\text{seq. length})}{\epsilon^2}\right)$ implies for any $\boldsymbol{\rho}$,
Average utility over training set is ϵ -close to **expected** utility

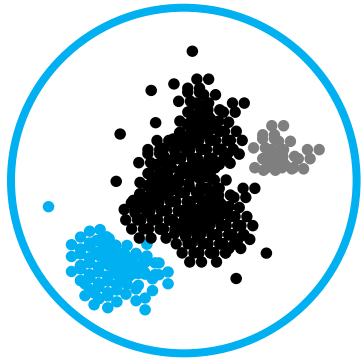


Outline

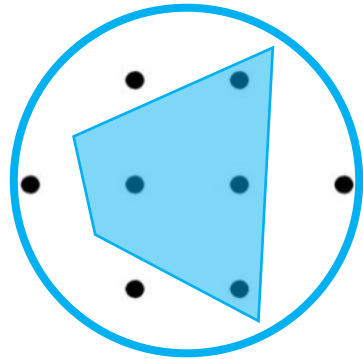
1. Introduction
2. Model and problem formulation
3. Our guarantees
- 4. Conclusions**

Conclusion

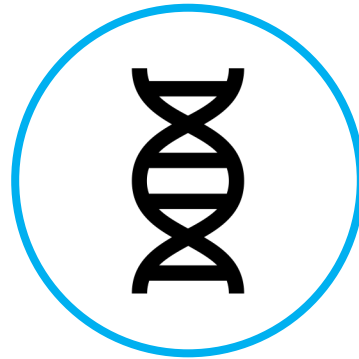
A **unifying** structure connects **seemingly disparate** problems:



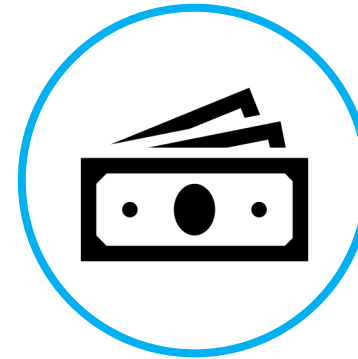
Clustering
algorithm
configuration



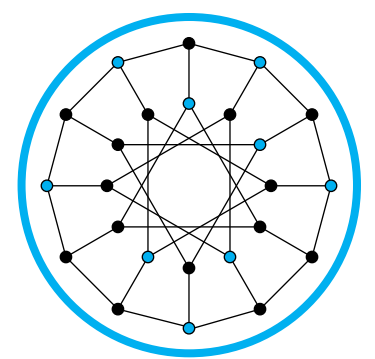
**Integer
programming**
algorithm
configuration



**Computational
biology**
algorithm
configuration



Mechanism
configuration



Greedy
algorithm
configuration

We use this structure to provide extremely **general** guarantees

How much data is sufficient to learn high-performing algorithms?

Ellen Vitercik
Carnegie Mellon University

STOC'21



Nina Balcan



Dan DeBlasio



Travis Dick



Carl Kingsford



Tuomas Sandholm