

# Parallel Machine Learning Algorithms in Bioinformatics and Global Optimization

Scott Clark<sup>1</sup>

Peter Frazier<sup>2</sup>

Zhong Wang<sup>3</sup>, Rob Egan<sup>3</sup>

Nick Hengartner<sup>4</sup>, Joel Berendzen<sup>4</sup>

<sup>1</sup>Cornell University Center for Applied Mathematics, Ithaca, NY

<sup>2</sup>Cornell University Department of ORIE, Ithaca, NY

<sup>3</sup>Joint Genome Institute, Walnut Creek, CA

<sup>4</sup>Los Alamos National Laboratory, Los Alamos, NM

May 4, 2012

# Outline

- ▶ Motivation
  - ▶ What is next-generation sequencing?

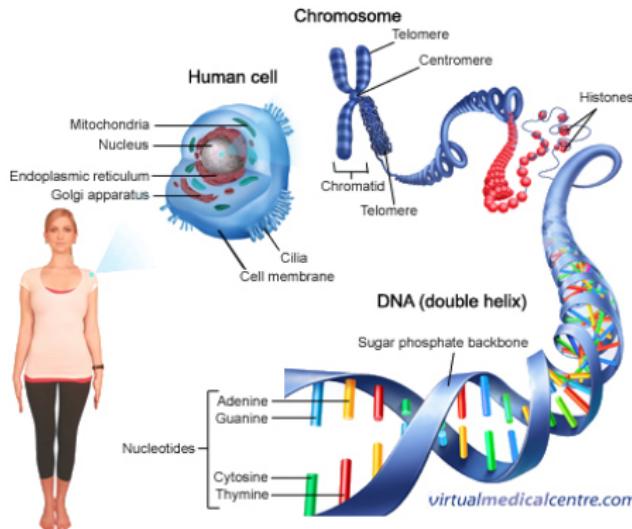
# Outline

- ▶ Motivation
  - ▶ What is next-generation sequencing?
- ▶ Validating Assemblies
  - ▶ What is an assembly?
  - ▶ Current methods
  - ▶ Using Bayes to build likelihoods
  - ▶ Dependence on assembly quality

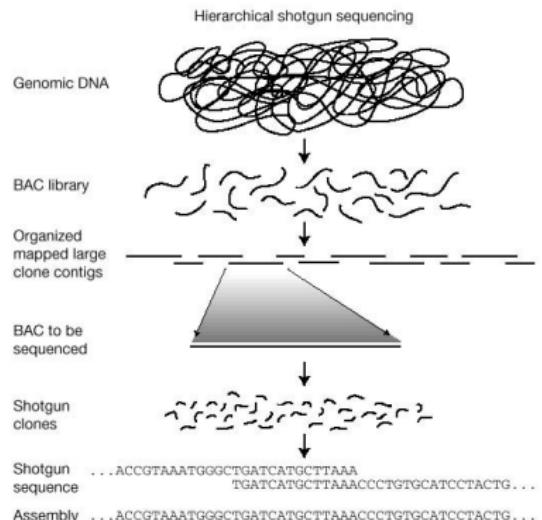
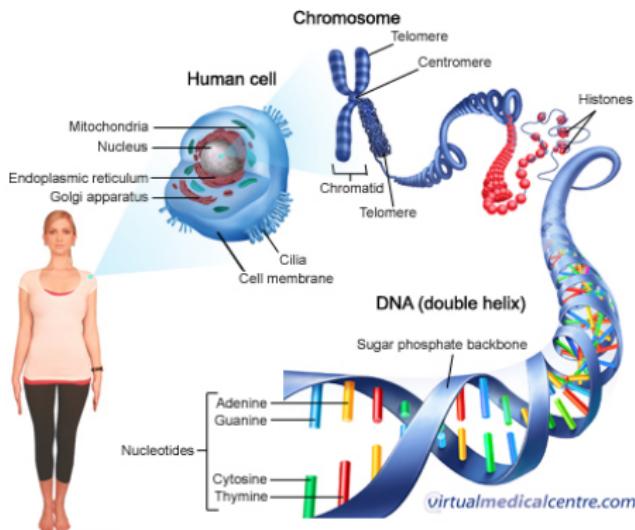
# Outline

- ▶ Motivation
  - ▶ What is next-generation sequencing?
- ▶ Validating Assemblies
  - ▶ What is an assembly?
  - ▶ Current methods
  - ▶ Using Bayes to build likelihoods
  - ▶ Dependence on assembly quality
- ▶ Expected Parallel Improvement
  - ▶ Parallel global optimization of expensive functions

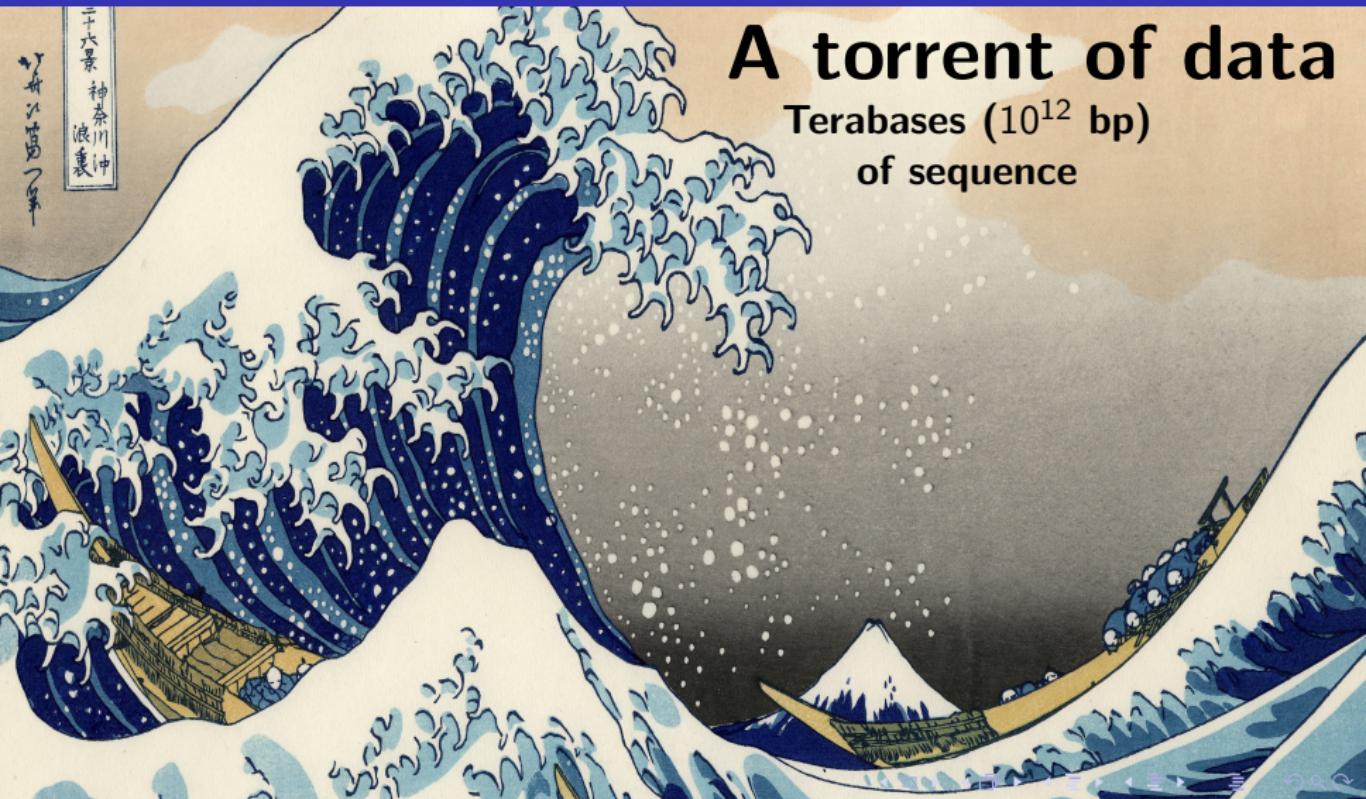
# What is next-generation sequencing?



# What is next-generation sequencing?

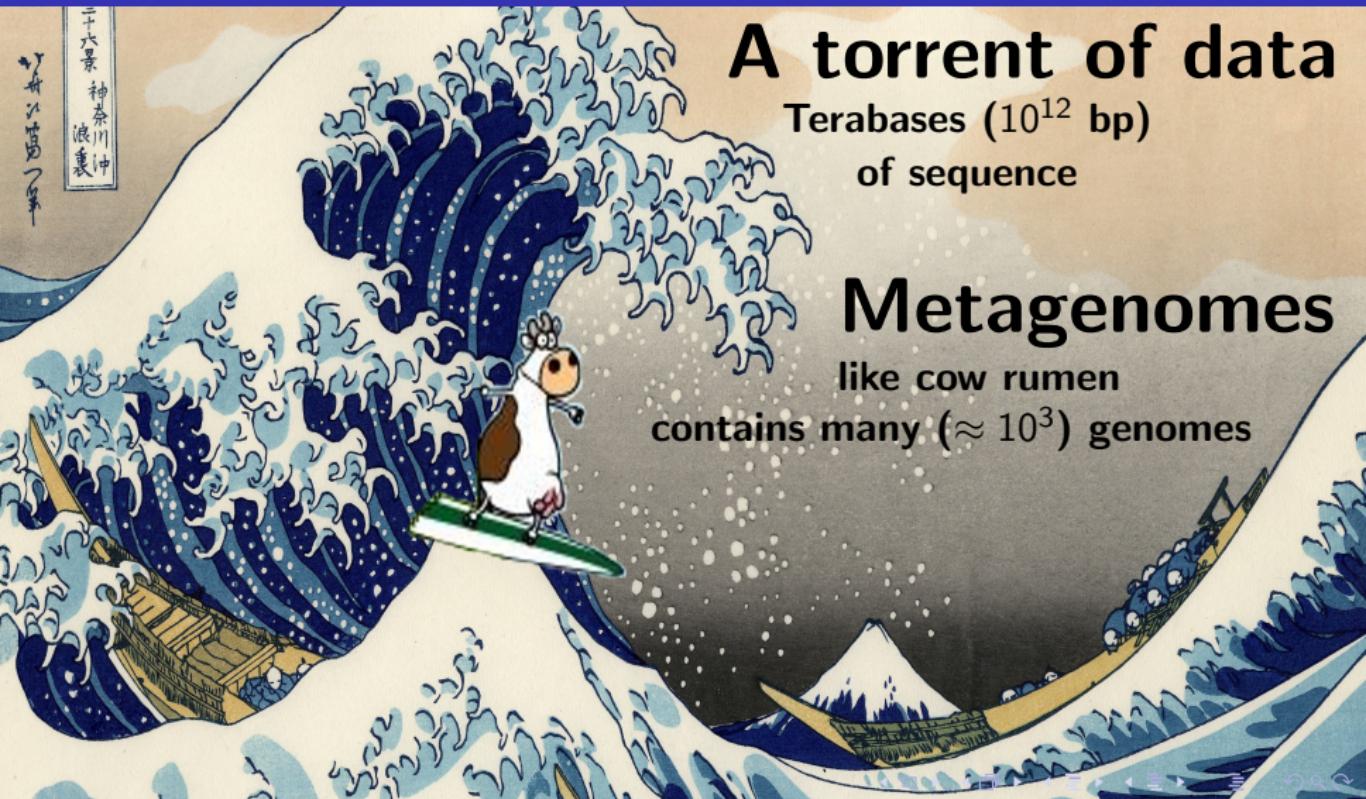


# What is the problem?

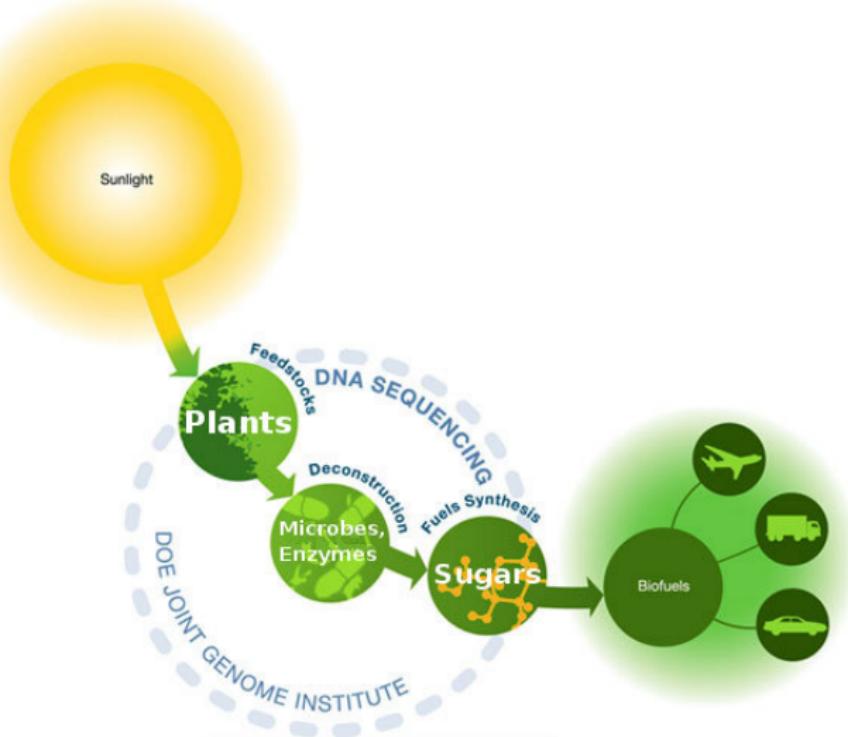


**A torrent of data**  
Terabases ( $10^{12}$  bp)  
of sequence

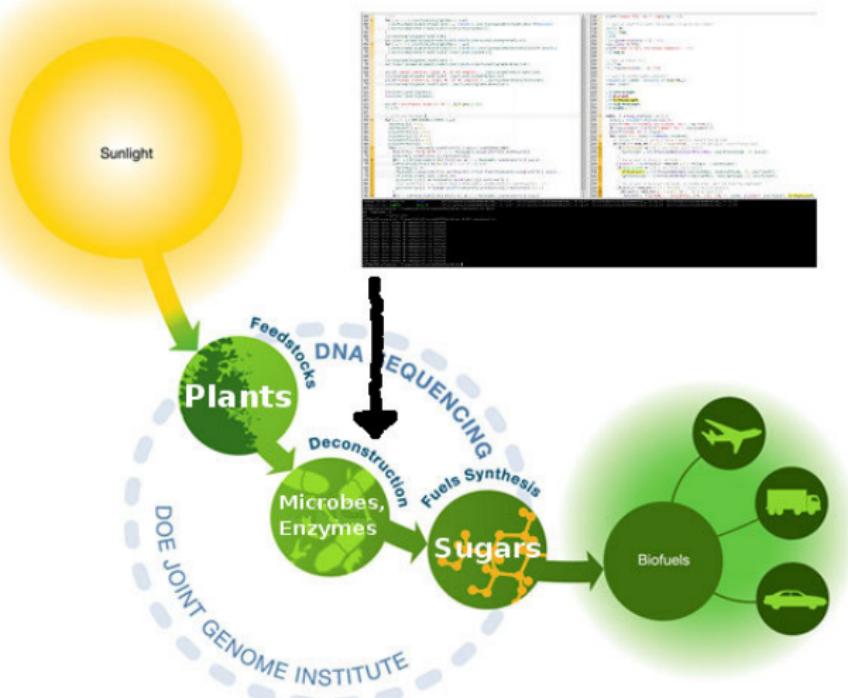
# What is the problem?



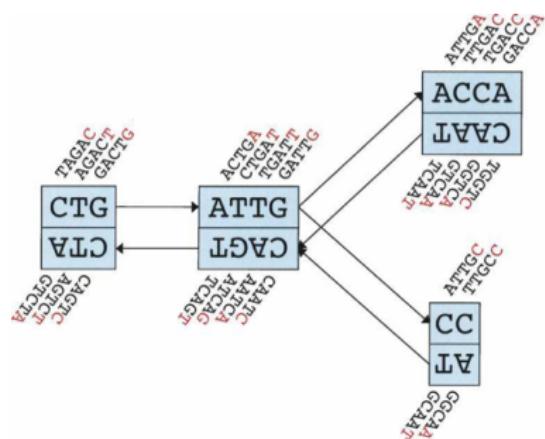
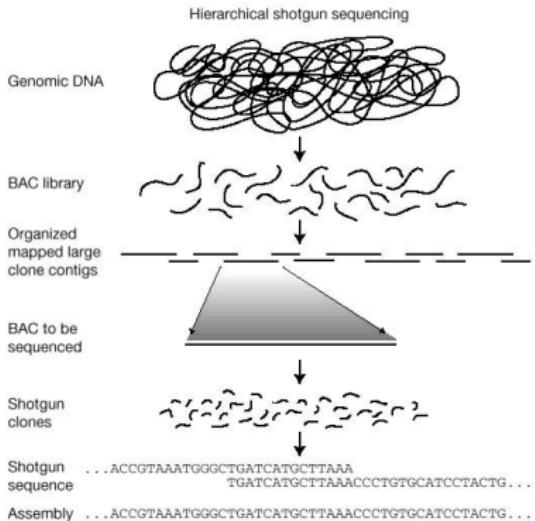
# Why do we (or the DOE) care?



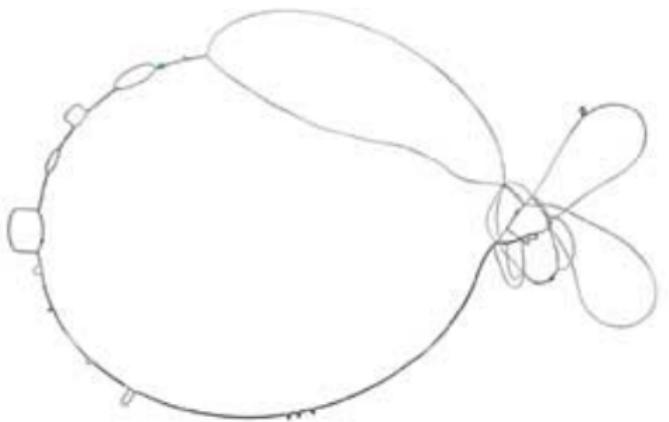
# Why do we (or the DOE) care?



# What is an assembly?



# What is an assembly?



# Current methods

Once we have an assembly from some method we want to know how good it is.

- ▶ Current Methods

1. Size of pieces (contigs), ie N50 size
2. Map it onto a known reference
3. Pathway analysis

# Current methods

Once we have an assembly from some method we want to know how good it is.

- ▶ Current Methods
  1. Size of pieces (contigs), ie N50 size
  2. Map it onto a known reference
  3. Pathway analysis
- ▶ Problems
  1. Accuracy irrelevant
  2. Sometimes there is no known reference
  3. Only works on specific pathway (with known reference)

# Using Bayes to build likelihoods

We want to be able to take an assembly and the reads that generated it and find a score, or how likely that assembly is,

$$P(\text{Assembly} | \text{All the Reads})$$

Which by Bayes rule is proportional to

$$P(\text{All the Reads} | \text{Assembly})$$

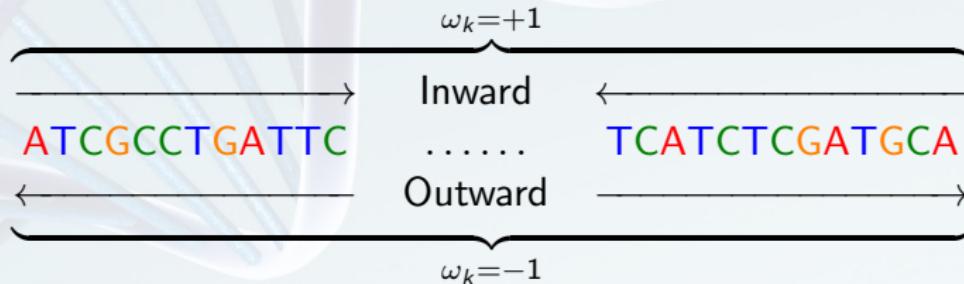
This will be our likelihood function.

## Read properties

**Mated Read  $R_k$  Properties**

Mate Pair #1:  $R_k^{(1)}$       Insert Area      Mate Pair #2:  $R_k^{(2)}$

Read  $R_k$ :   
length:  $L(R_k^{(1)}) \sim \Theta_{L_k}$       length:  $I_k \sim \Theta_{I_k}$       length:  $L(R_k^{(2)}) \sim \Theta_{L_k}$

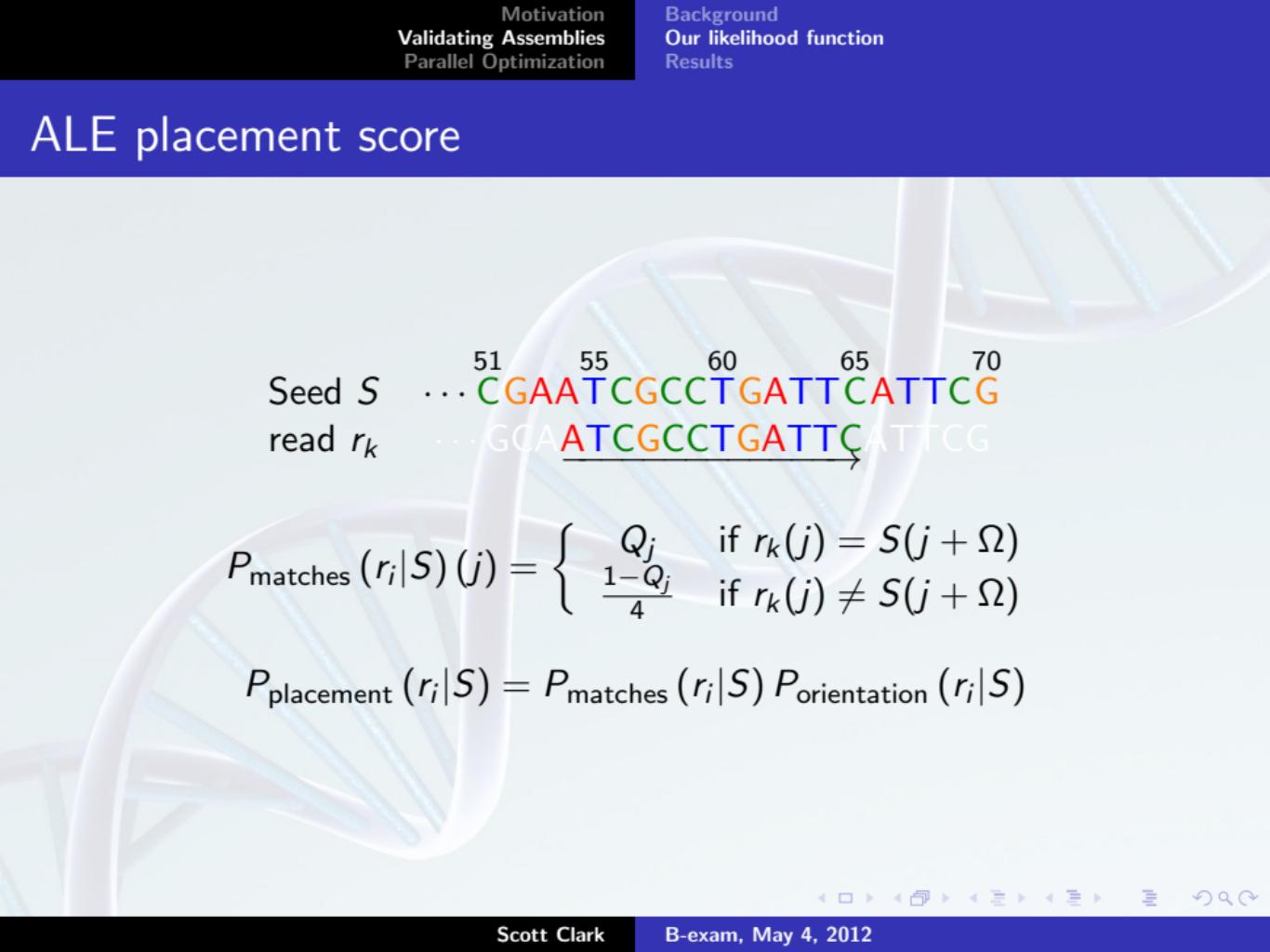
**Orientation  $\omega_k$  of mated read  $R_k$** 

# Components of the likelihood function

The Likelihood needs to take at least the following into account:

- ▶ Read **sequences** need to agree with assembly.
- ▶ **Insert lengths** must be consistent.
- ▶ **Orientation** of mated reads needs to be consistent.
- ▶ **Number of reads mapping** onto the assembly.
- ▶ **Coverage** normally distributed.
- ▶  **$k$ -mer frequency** must be consistent. (within a contig)

## ALE placement score



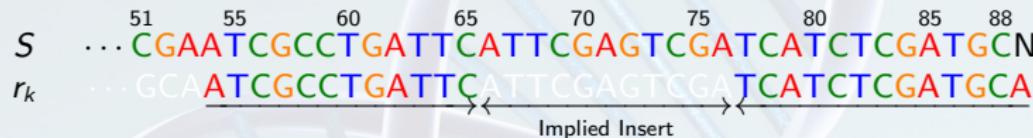
Seed  $S$     ... **CGAATCGCCTGATTCAATT**CG  
read  $r_k$     ... GCA**ATCGCCTGATT**CATT $\rightarrow$

$$P_{\text{matches}}(r_i|S)(j) = \begin{cases} Q_j & \text{if } r_k(j) = S(j + \Omega) \\ \frac{1-Q_j}{4} & \text{if } r_k(j) \neq S(j + \Omega) \end{cases}$$

$$P_{\text{placement}}(r_i|S) = P_{\text{matches}}(r_i|S) P_{\text{orientation}}(r_i|S)$$

# ALE insert score

Total ALE score needs to take insert size into account



$$P_{\text{insert}}(r_i|S) = \text{Normal}(L_i; \mu, \sigma^2)$$

# ALE depth score

We expect the coverage of the reads when mapped back onto the seeds to be Poisson distributed (Lander 1988)

$$P_{\text{depth}}(d_j|S, X_i) = \text{Poisson}(d_j; Y_i)$$

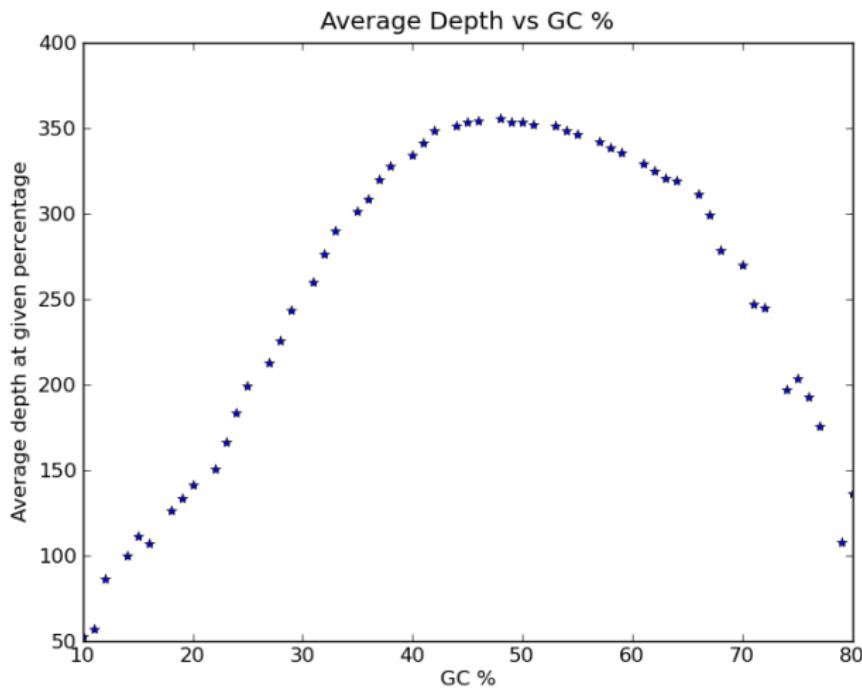
$$Y_i \sim \text{Gamma}(\max(10, \mu_{\text{depth}(X_i)}), 1)$$

$$P_{\text{depth}}(d_j|S, X_i) = \text{NegBinom}(d_j; \max(10, \mu_{\text{depth}(X_i)}), 1/2)$$

## GC correction

- ▶ The sequencing technology introduces a “GC bias”
- ▶ Those bases are overrepresented in the data
- ▶ We compute 100 different Poisson distributions based on the % of GC in each read

# ALE depth score: GC bias



# ALE kmer score

What are  $k$ -mers?

- ▶ Contiguous subsequences of length  $k$
- ▶ The frequency a  $k$ -mer appears (for  $k \geq 4$ ) is relatively distinct between species
- ▶ We can use this information to distinguish individual species from a metagenome

$$f_i = \frac{n_i}{\sum_{j \in K} n_j}$$

$$P_{kmer}(S) = \prod_{i \in K} f_i^{n_i}$$

# Normalization and total scores

The total ALE score

$$P(S|R) = \frac{P(R|S)P(S)}{Z}$$

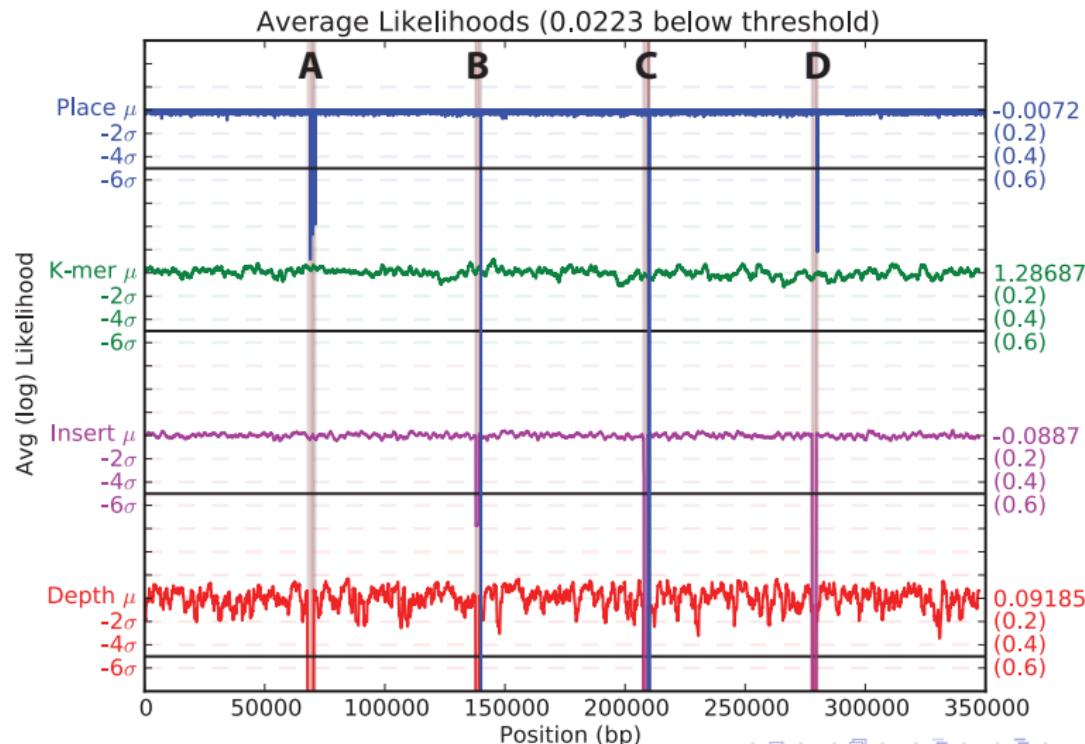
$$P(R|S) = P_{\text{placement}}(R|S)P_{\text{insert}}(R|S)P_{\text{depth}}(R|S)$$

$$P(S) = P_{\text{kmer}}(S)$$

Comparing two assemblies

$$A_1 - A_2 = \log \left( \frac{P(R|S_1)P(S_1)}{P(R|S_2)P(S_2)} \right)$$

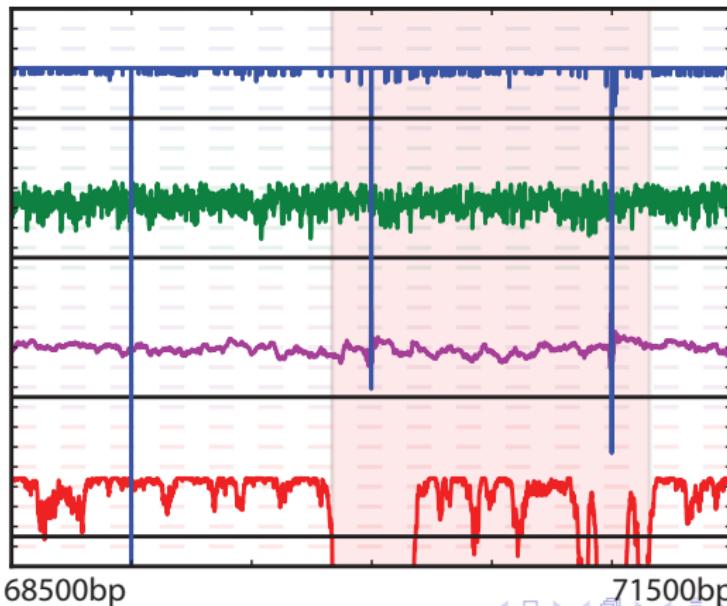
# Likelihood with permutation from a known assembly



# Likelihood with permutation from a known assembly

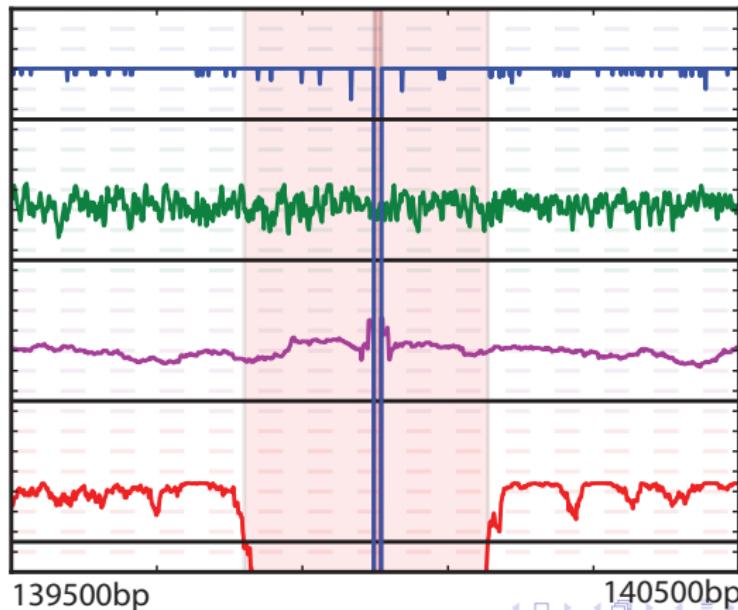
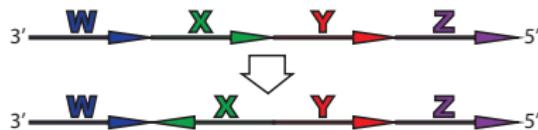
A

...ATCG**G**GAT... ...ATCG**G**GAT... ...ATCGAT...  
↓ ↓ ↓  
...ATCT**T**GAT... ...ATCGAT... ...ATCT**T**GAT...



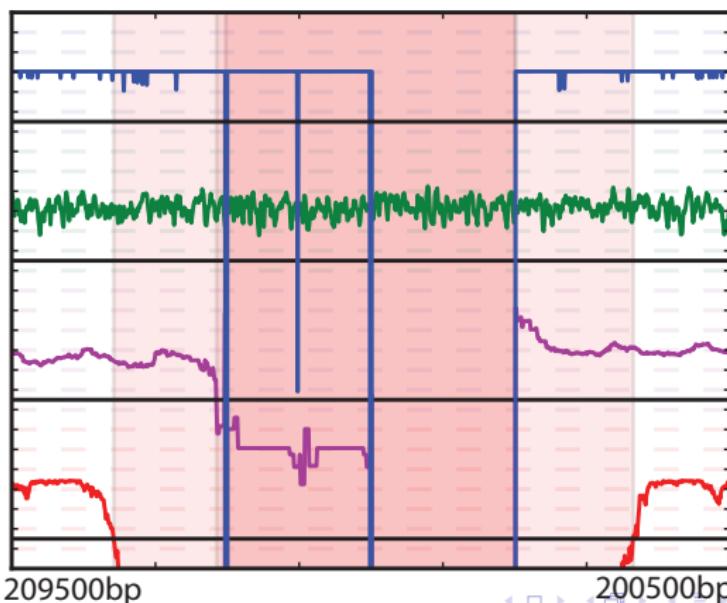
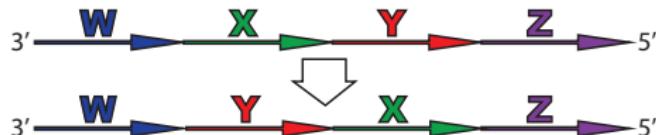
# Likelihood with permutation from a known assembly

**B**



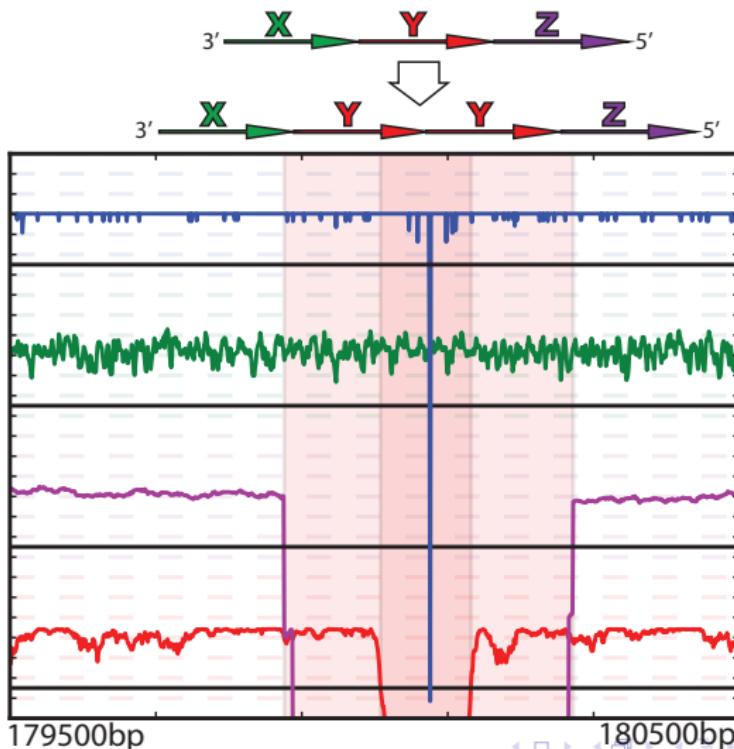
# Likelihood with permutation from a known assembly

C

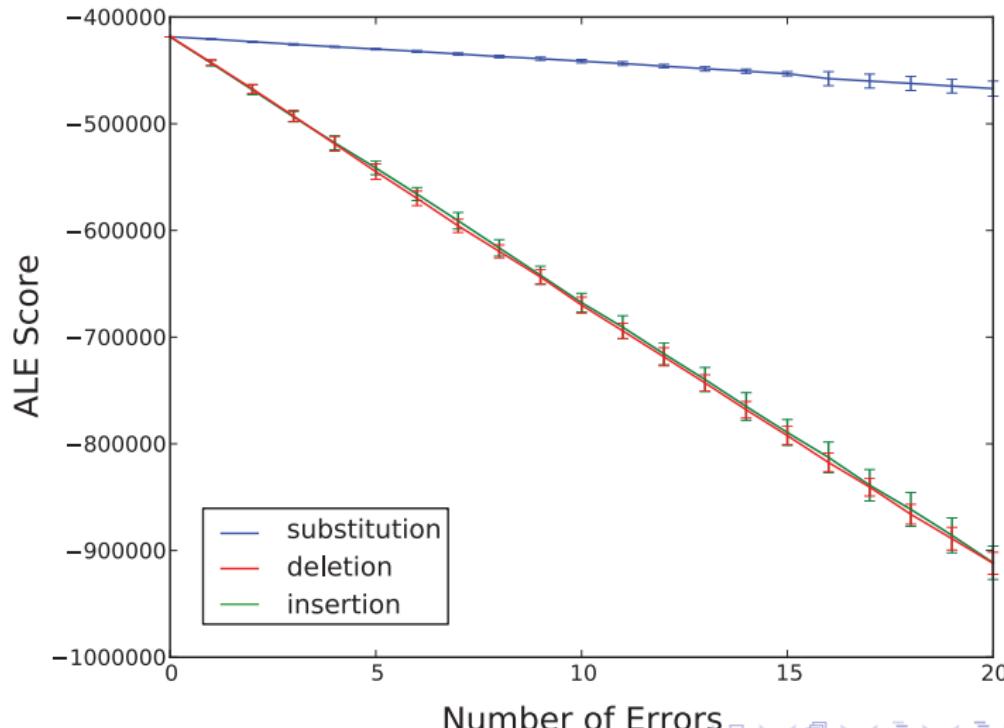


# Likelihood with permutation from a known assembly

D

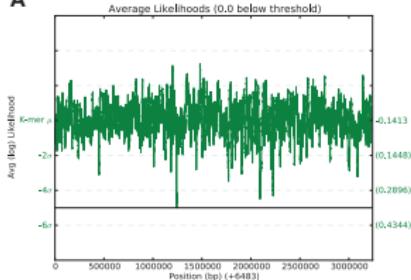


# Likelihood vs # permutations from a known assembly

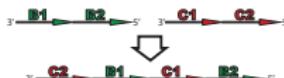
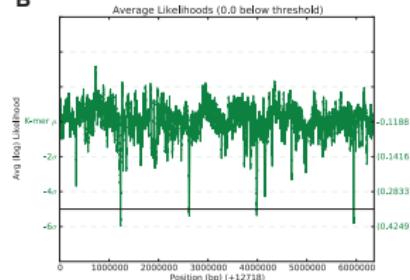


# Detecting chimeric assemblies in a synthetic metagenome

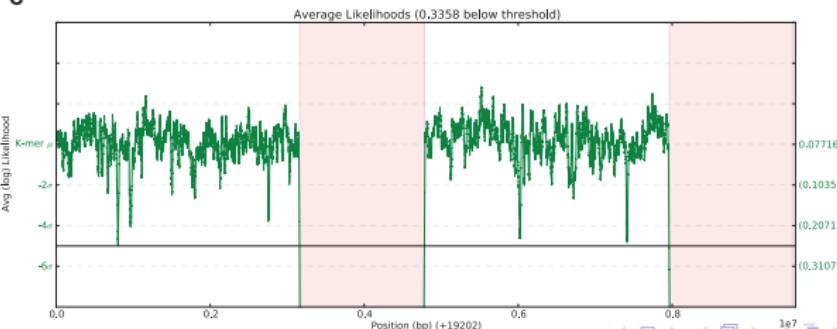
A



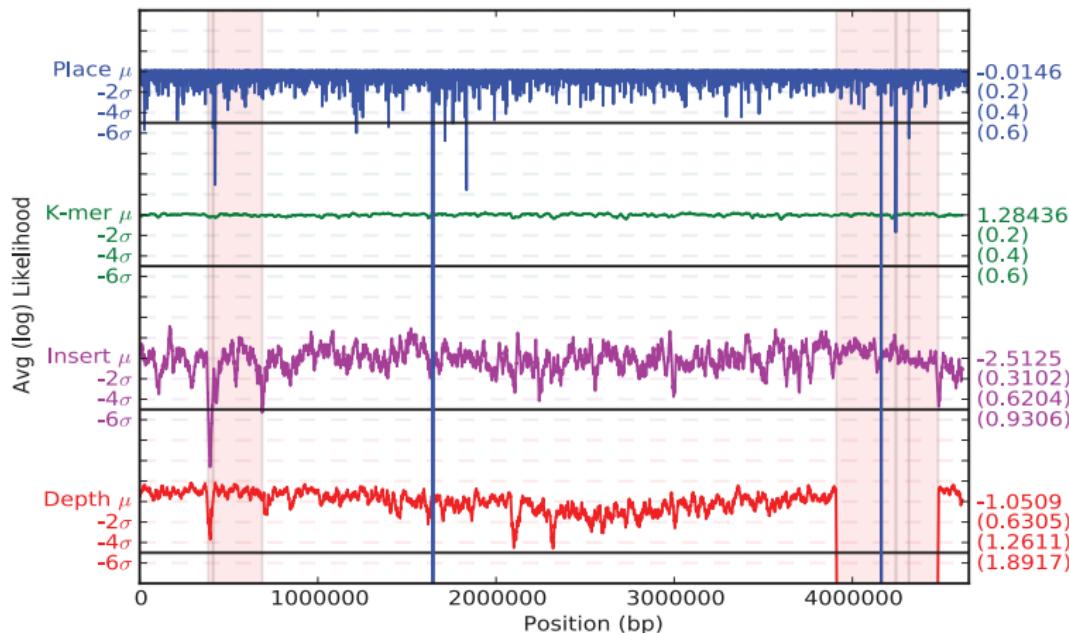
B



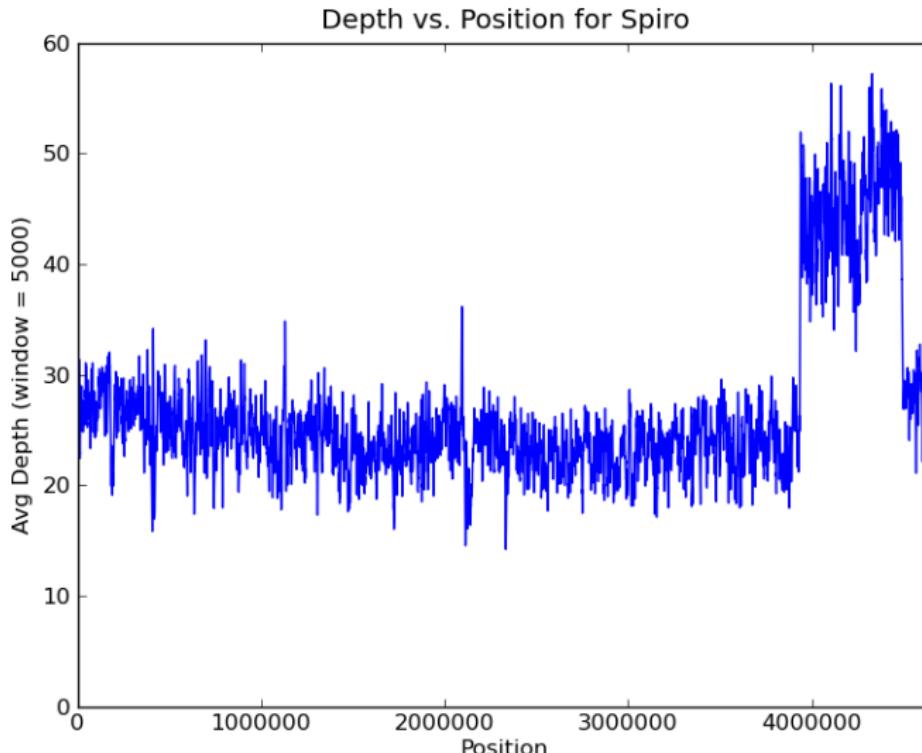
C



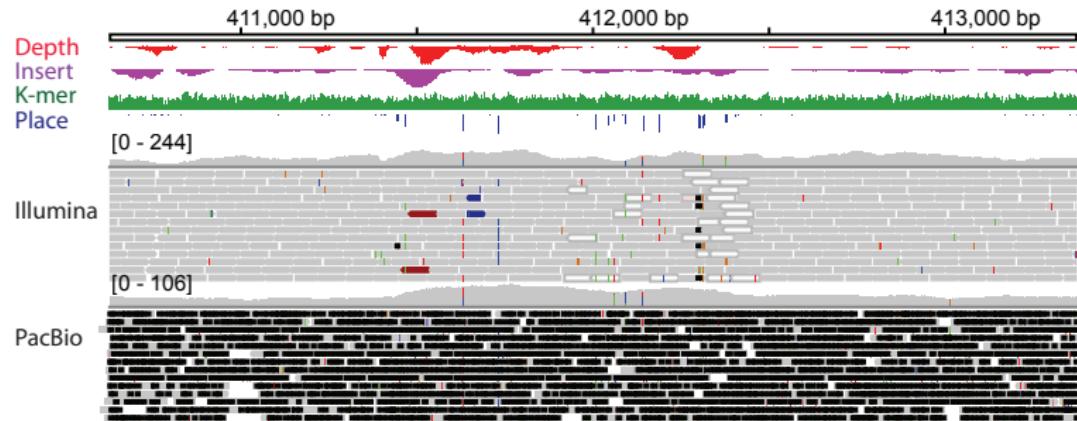
## Discovery of errors in real genome assemblies



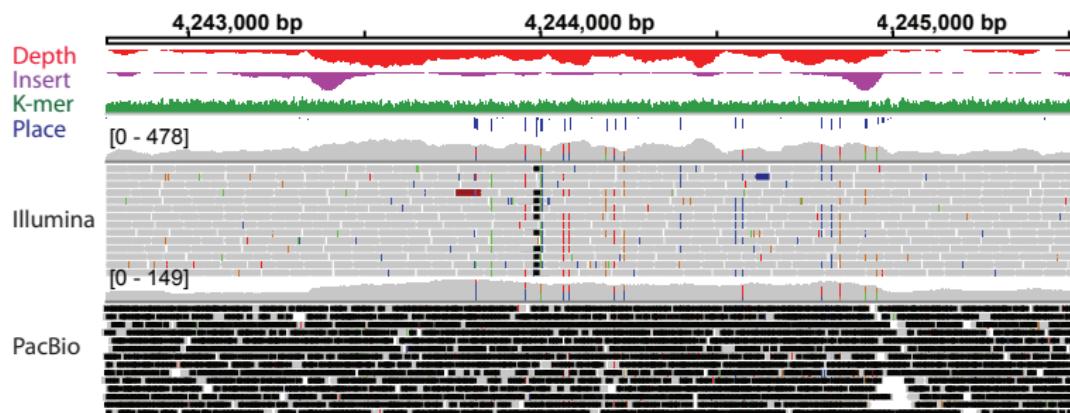
# Discovery of errors in real genome assemblies



# Discovery of errors in real genome assemblies



# Discovery of errors in real genome assemblies



## Performance with Pacific Biosciences RS data

Operation Type	Mutation Details	Position	548x	50x	548x
			Eviron (PacBio)	Rank (ALE)	Rank (ALE)
Sub	C → A	881	1	5	5
Ins	CC → CCC	2161	-	14	12
Del	G → -	3681	1	9	8
N/A	-	15712	-	7	-
Del	AACGGGCAGA	16561	1	4	4
Ins	AACGGGCAGA	17030	1	3	2
Sub	A → G	22881	1	10	7
Sub	T → A	28561	1	11	10
Del	T	34560	1	12	11
Sub	G → C	36560	1	8	9
Ins	ACGTACGT	40721	1	1	1
N/A	-	41318	-	13	-
Del	TCATCGCG	43200	-	6	6
Ins	C	47600	1	2	3

# ALE Conclusion

In ALE we have developed an algorithm and software package that

- ▶ Can validate assemblies in a rigorous and probabilistic way\*
- ▶ Allows for the quick discovery of errors in assemblies
- ▶ Can validate metagenomic assemblies and datasets\*
- ▶ Can be used to help “finish” genomes
- ▶ Open source, easy to install, easy to fit into a pipeline

# Expected Parallel Improvement

We want a fast/efficient, derivative-free and global optimization algorithm for an expensive to evaluate, “black box” function  $f$ .

$$\max_{x \in A} f(x).$$

The function  $f$  has domain  $A \subseteq \mathbb{R}^d$ , possibly constrained.

This function can be anything, like an assembly likelihood given either the reference or some parameters for an assembler.

# The math (figures next!)

We begin with a Gaussian process prior on a continuous function  $f$ .

$$f \sim \text{GP}(\mu(\cdot), \Sigma(\cdot, \cdot))$$

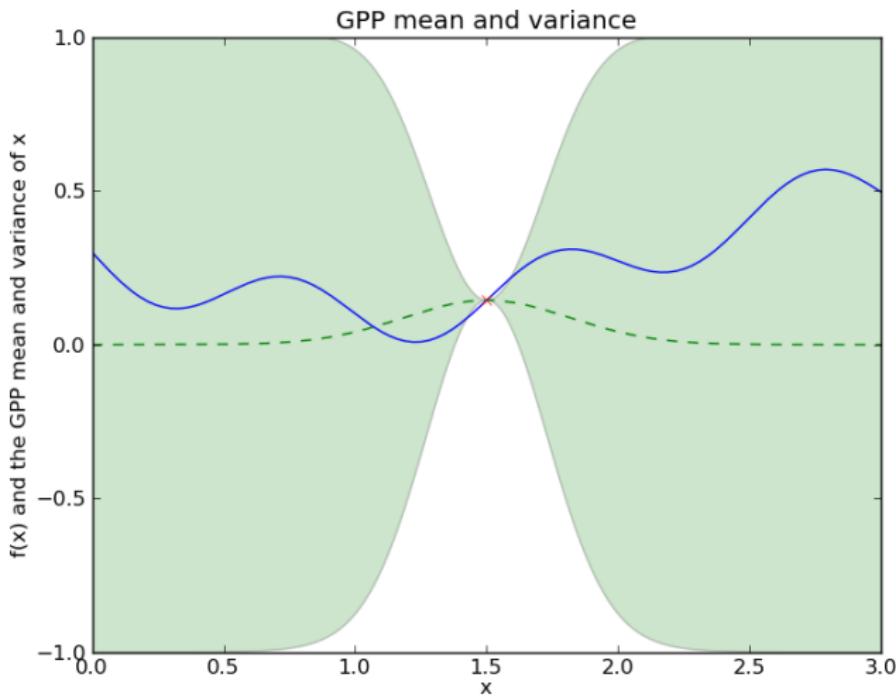
$$(f(x_1), \dots, f(x_n)) \sim N \left( \begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} \Sigma(x_1, x_1) & \cdots & \Sigma(x_n, x_1) \\ \vdots & \ddots & \vdots \\ \Sigma(x_1, x_n) & \cdots & \Sigma(x_n, x_n) \end{bmatrix} \right)$$

$$\mu_n(x_\star) = K(\vec{x}_\star, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\vec{y}$$

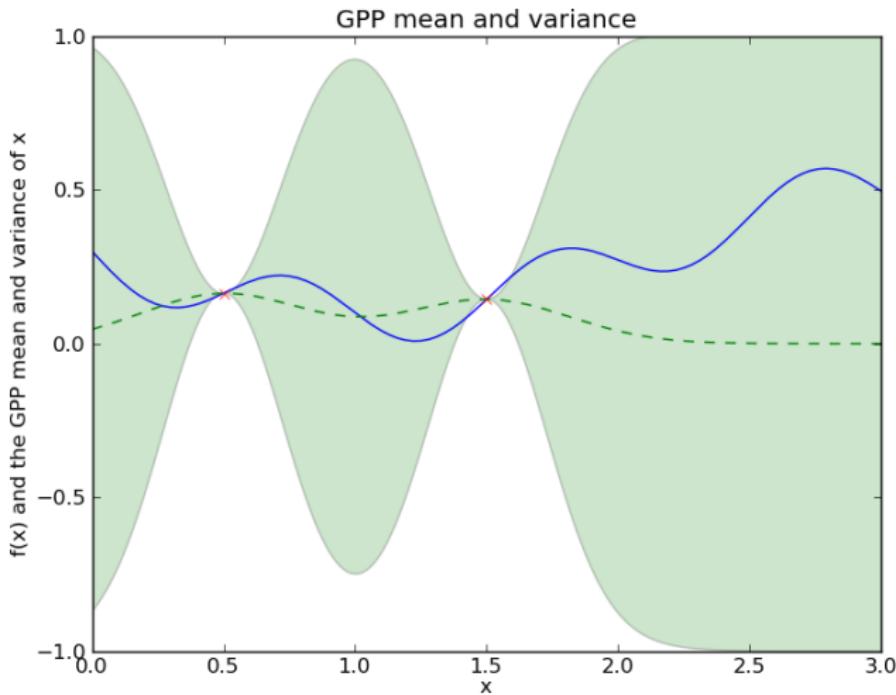
$$\Sigma_n(x_\star) = K(\mathbf{x}_\star, \mathbf{x}_\star) - K(\mathbf{x}_\star, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{x}_\star)$$

$$K(\vec{y}, \vec{z})_{ij} = \text{cov}(y_i, z_j)$$

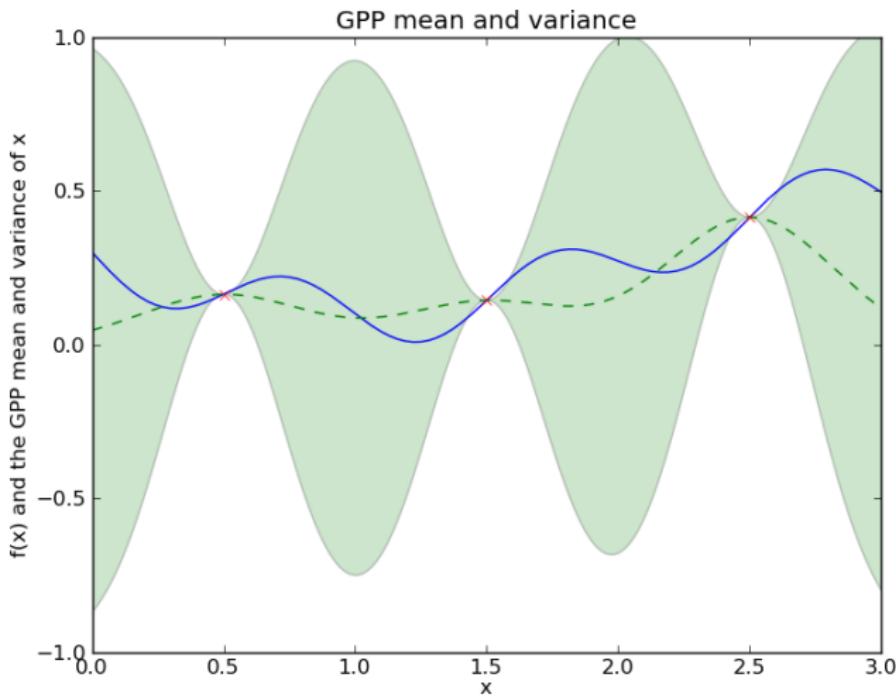
# GPP evolving with information



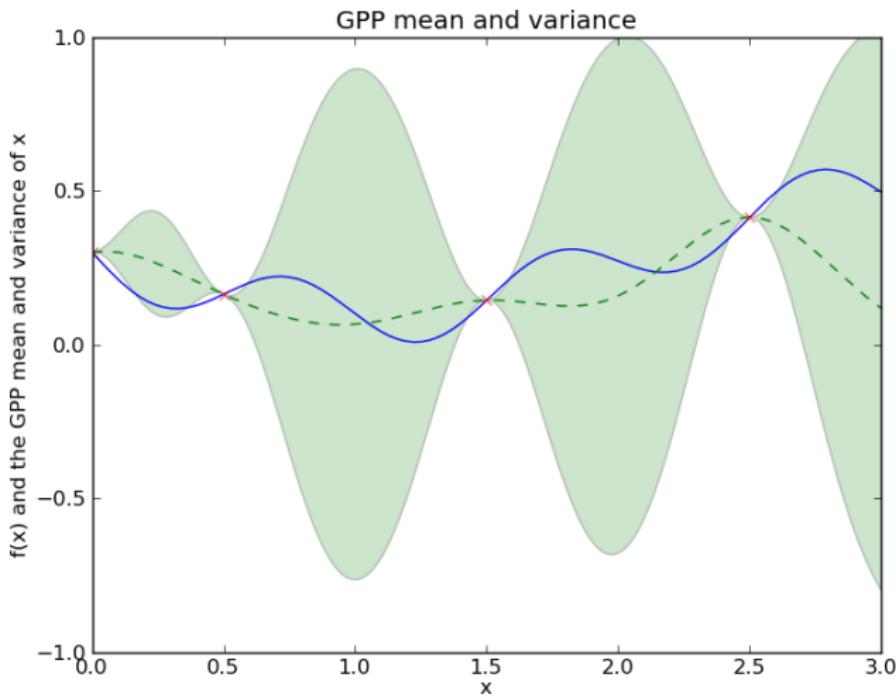
# GPP evolving with information



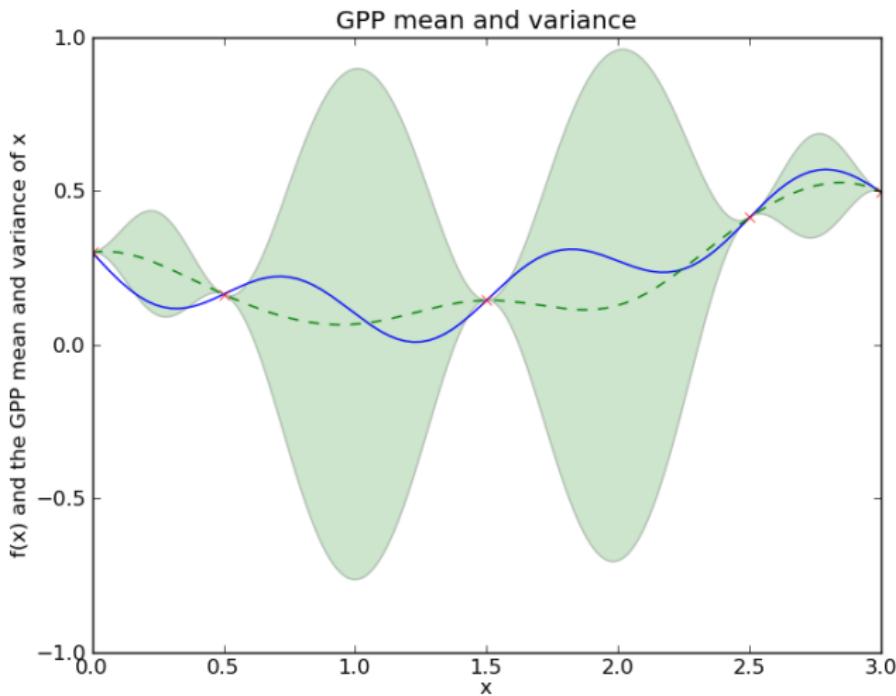
## GPP evolving with information



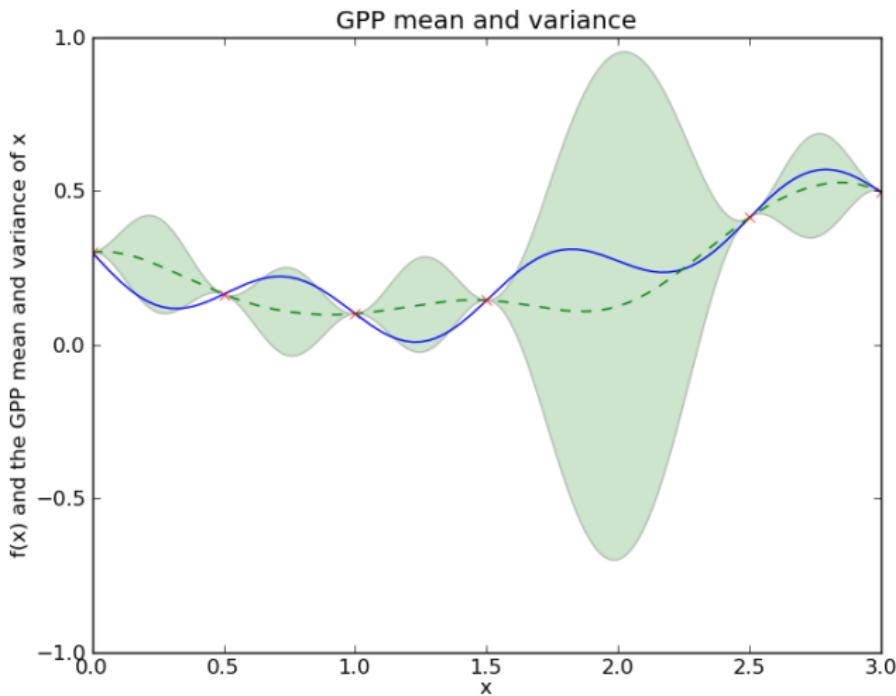
# GPP evolving with information



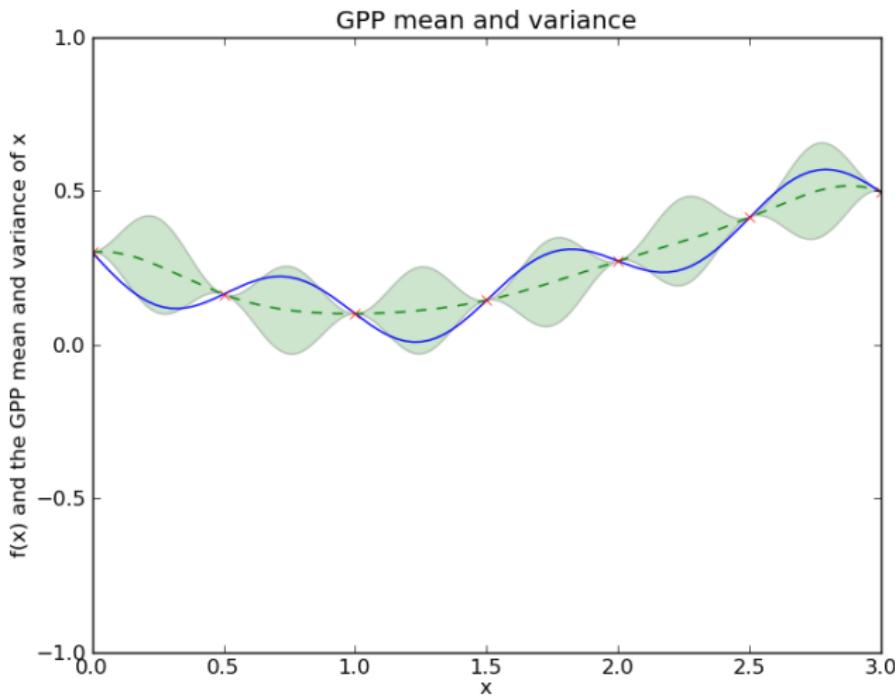
## GPP evolving with information



## GPP evolving with information



# GPP evolving with information



# Expected Improvement

$$\text{EI}(x) = \mathbb{E}_n \left[ [f(x) - f_n^*]^+ \right] = \mathbb{E} \left[ f_{n+1}^* - f_n^* | x_n = x \right]$$

where  $f_n^* = \max_{m \leq n} f(x_m)$ .

The core of this idea is that we can calculate the expected improvement for simultaneous evaluation of points

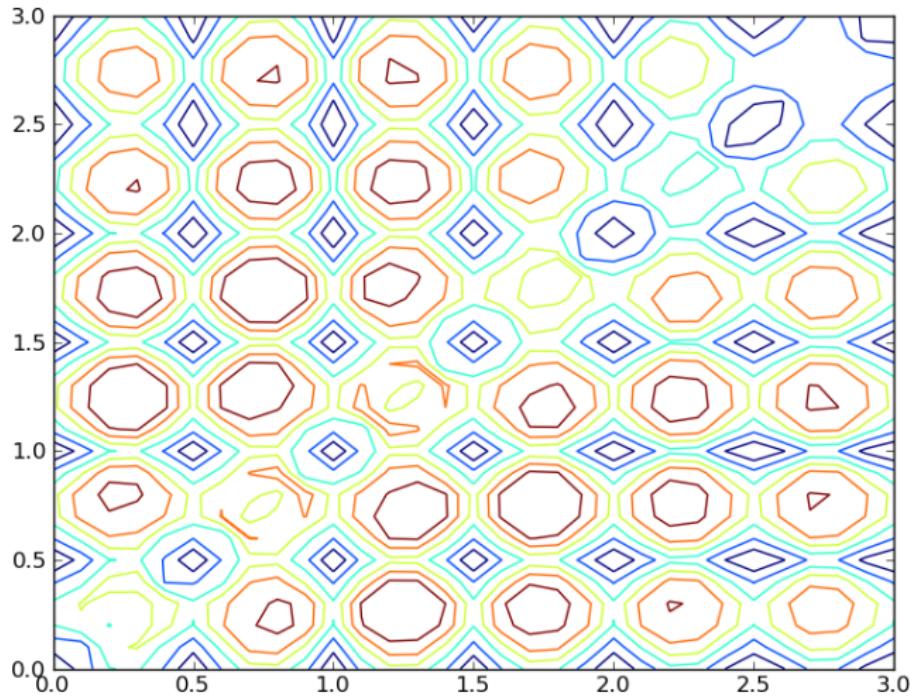
$x_{n+1}, \dots, x_{n+l} = \vec{x}$  as

$$\text{EI}(x_{n+1}, \dots, x_{n+l}) = \mathbb{E}_n \left[ [\max \{f(x_{n+1}), \dots, f(x_{n+l})\} - f_n^*]^+ \right].$$

The optimization then approximates the solution to

$$\operatorname{argmax}_{\vec{x} \in \mathbb{R}^{d \times l}} \text{EI}(\vec{x}).$$

# Expected Improvement 2-D



# Expected Improvement 2-D

To optimize  $\text{EI}(\vec{x})$ , we calculate stochastic gradients

$$g(\vec{x}) = \nabla \text{EI}(\vec{x})$$

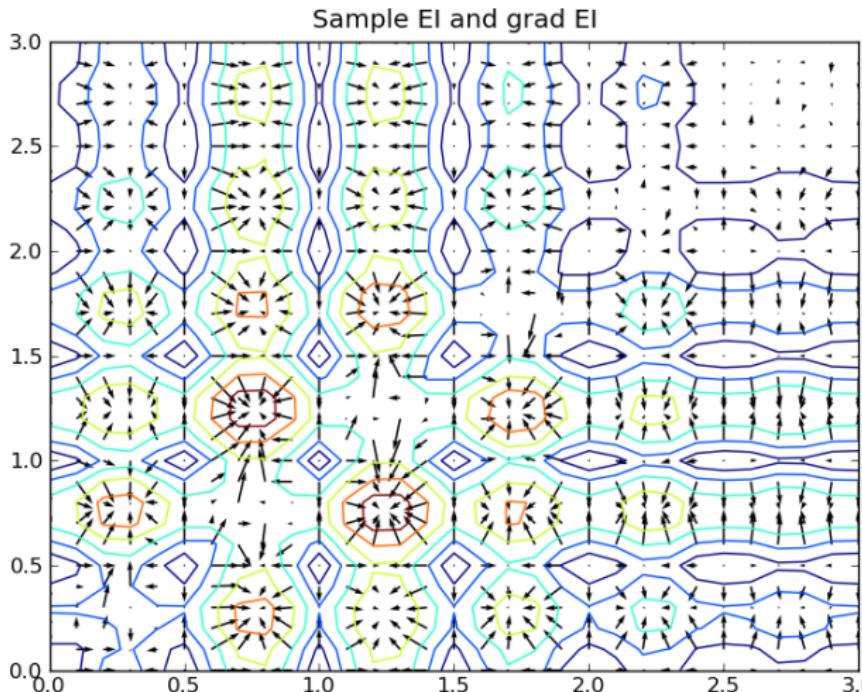
Which results in (skipping some derivations for the sake of time/sanity)

$$\frac{\partial}{\partial x_{\star t}} \mu_{\star i} = \begin{cases} \sum_{j=1}^N (K^{-1} \vec{y})_j \frac{\partial}{\partial x_{\star i}} \text{cov}(x_{\star i}, X_j) & \text{for } i = t \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial x_{\star t}} \Sigma_{ij} = \begin{cases} 2 \sum_{p=1}^N \sum_{q=1}^N K_{qp}^{-1} \left( \text{cov}(x_{\star i}, X_q) \frac{\partial}{\partial x_{\star i}} \text{cov}(x_{\star i}, X_p) \right) & t = i = j \\ \sum_{p=1}^N \sum_{q=1}^N K_{qp}^{-1} \text{cov}(x_{\star j}, X_p) \frac{\partial}{\partial x_{\star i}} \text{cov}(x_{\star i}, X_q) & t = i \neq j \\ \sum_{p=1}^N \sum_{q=1}^N K_{qp}^{-1} \text{cov}(x_{\star i}, X_p) \frac{\partial}{\partial x_{\star j}} \text{cov}(x_{\star j}, X_q) & t = j \neq i \\ 0 & \text{otherwise} \end{cases}$$

Which we use to inform the algorithm for the differentiation of the Cholesky decomposition... (see code/paper)

# Stochastic Gradient of the Expected Improvement using MC



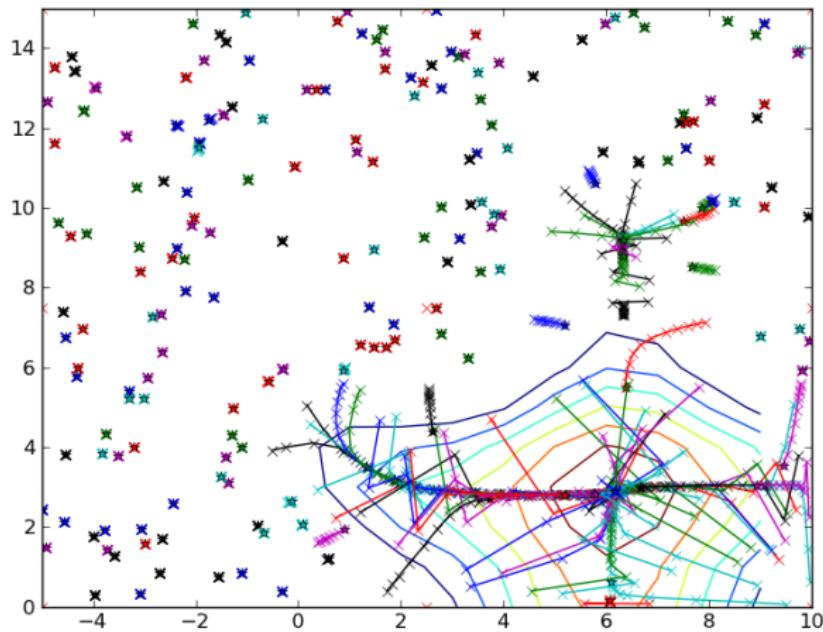
# The Algorithm

## The Algorithm

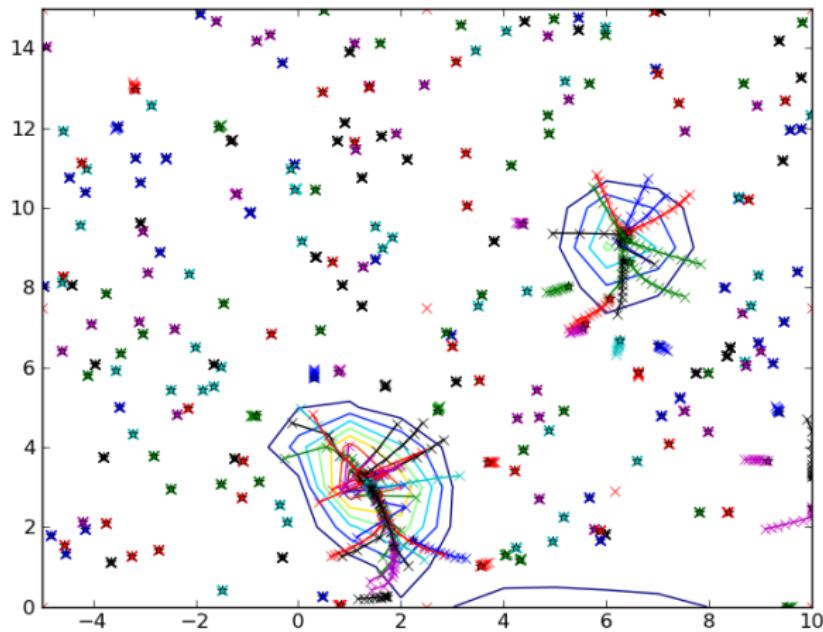
1. Choose the set of points with the highest EI
2. As nodes return values search for new best EI along the space constrained by running simulations

$$\vec{x}_i^{(t+1)} = \vec{x}_i^{(t)} + \frac{a}{t^\gamma} \nabla_{\vec{x}_i} \text{EI} \left( \vec{P}^{(t)} | \vec{X} \right)$$
$$\left| \vec{x}_i^{(t+1)} - \vec{x}_i^{(t)} \right| < \epsilon$$

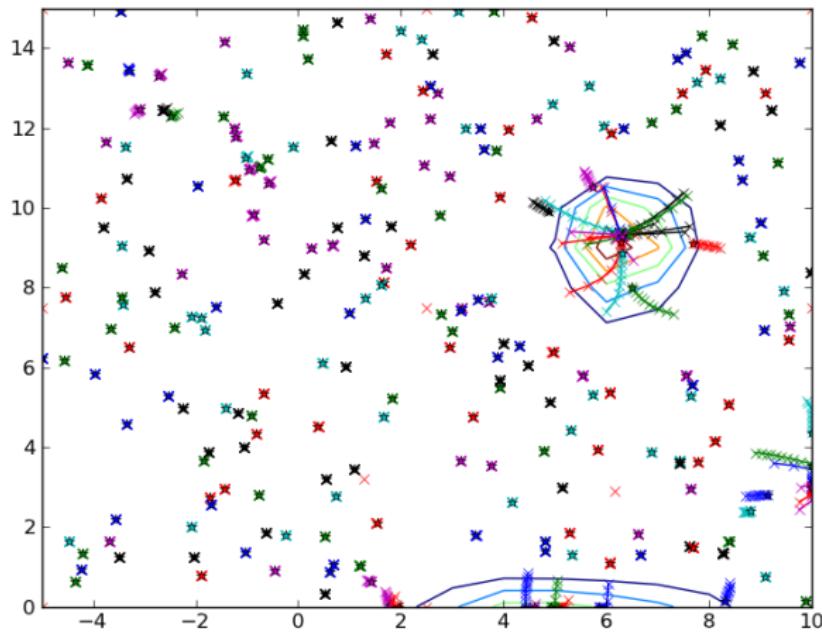
# Multistart paths (1/5)



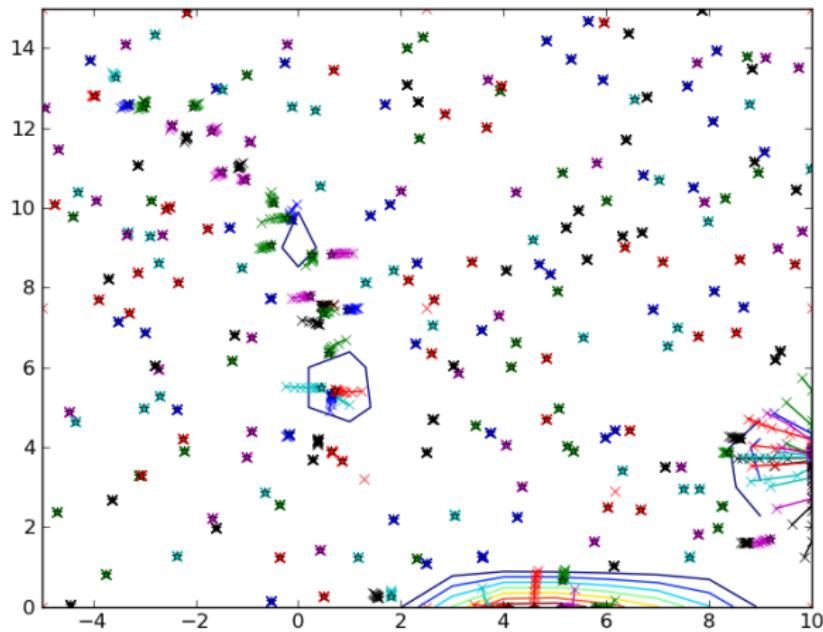
## Multistart paths (2/5)



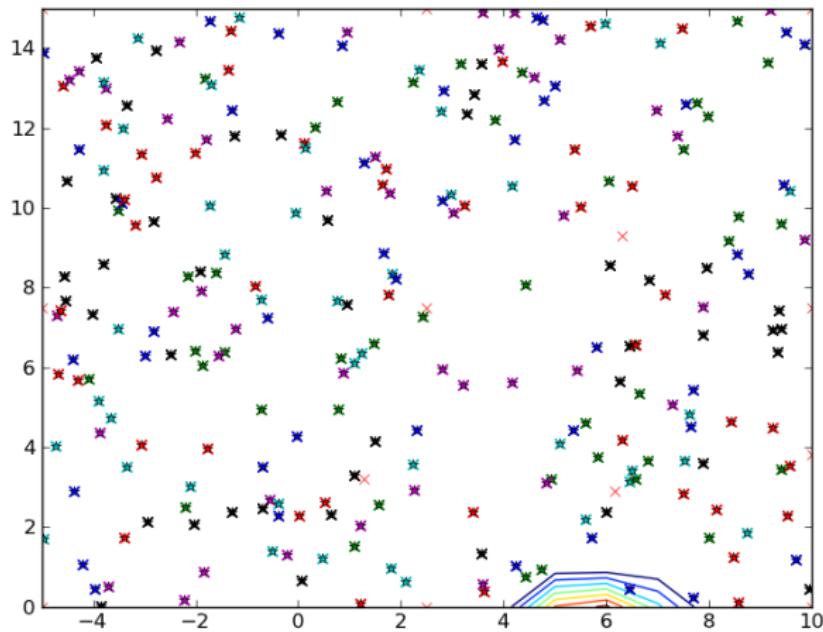
## Multistart paths (3/5)



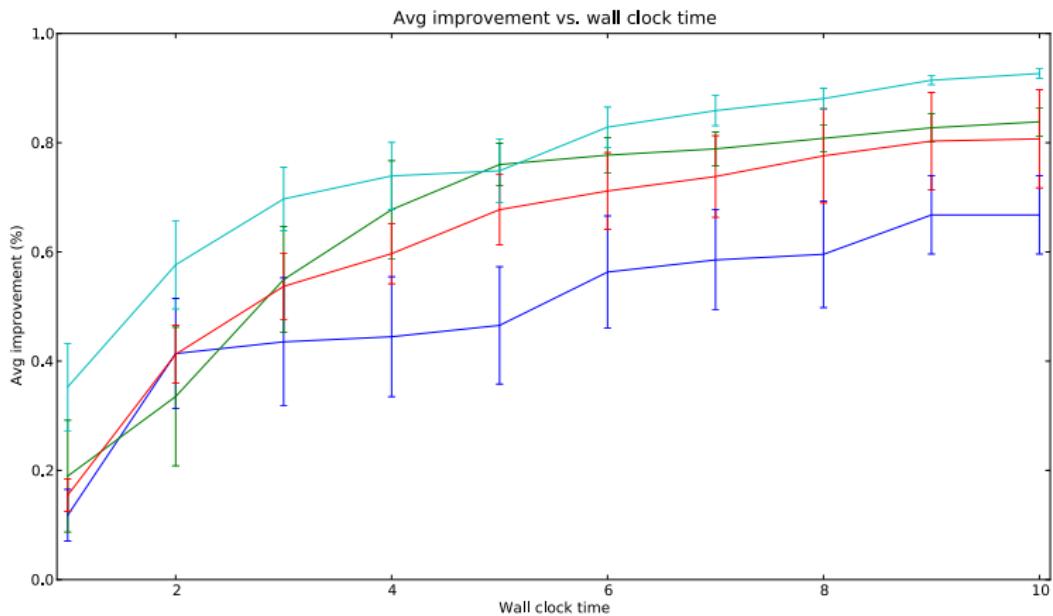
## Multistart paths (4/5)



## Multistart paths (5/5)



## Speedup



# EPI Conclusion

WHAT WE DID

# Velvetrope (Part III of thesis)

Developed an algorithm and software package that

- ▶ Finds motifs in sequences using a novel statistical model
- ▶ Is faster than the current methods (GPU implementation, CUDA)
- ▶ Can handle cases other methods cannot
- ▶ Natively allows for quick analysis of aggregate data
- ▶ Is open source and easy to fit into a pipeline

# Project status

ALE	Software: <a href="https://github.com/sc932/ALE">github.com/sc932/ALE</a> (UoI/CNSA open source license) Presented: SC11, SIAM CSE11, DOE CSGF11 INFORMS10, JGI Seminar Paper: submitted to Bioinformatics
EPI	Software: <a href="https://github.com/sc932/EPI">github.com/sc932/EPI</a> (UoI/CNSA open source license) Presented: SC12, DOE CSGF12 Paper: in preparation
Velvetrope	Software: <a href="https://github.com/sc932/Velvetrope">github.com/sc932/Velvetrope</a> (GPL v2 open source license) Presented: SC10, DOE CSGF10 Paper: profiled in DIEXIS

# Acknowledgements



- ▶ DOE Computational Science Graduate Fellowship
- ▶ Advisor: Peter Frazier (Cornell ORIE)
- ▶ DOE Joint Genome Institute (JGI) Genome Analysis Group
  - ▶ Zhong Wang (Mentor)
  - ▶ Rob Egan (Co-Mentor)
- ▶ Los Alamos National Laboratory (LANL) Metagenomics Group
  - ▶ Nick Hengartner (Mentor)
  - ▶ Joel Berendzen (Co-Mentor)
- ▶ Committee
  - ▶ Steve Strogatz (Math)
  - ▶ Bart Selman (CS)
  - ▶ Jim Renegar (Math proxy)

DOE Grant:  
DE-FG02-97ER25308  
DE-AC02-05CH11231