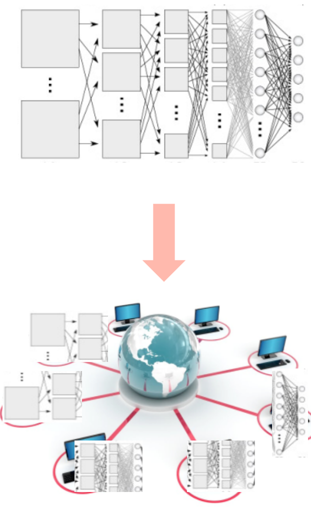
School of Computer Science
Carnegie Mellon

Probabilistic Graphical Models

Approximate Inference: Parallel MCMC



Eric Xing

Lecture XX, April 23rd, 2014

Recap of MCMC

- Markov Chain Monte Carlo methods use adaptive proposals Q(x'|x) to sample from the true distribution P(x)

- Metropolis-Hastings allows you to specify any proposal Q(x'|x)
  - But choosing a good Q(x'|x) requires care

- Gibbs sampling sets the proposal Q(x'|x) to the conditional distribution P(x'|x)
  - Acceptance rate always 1!

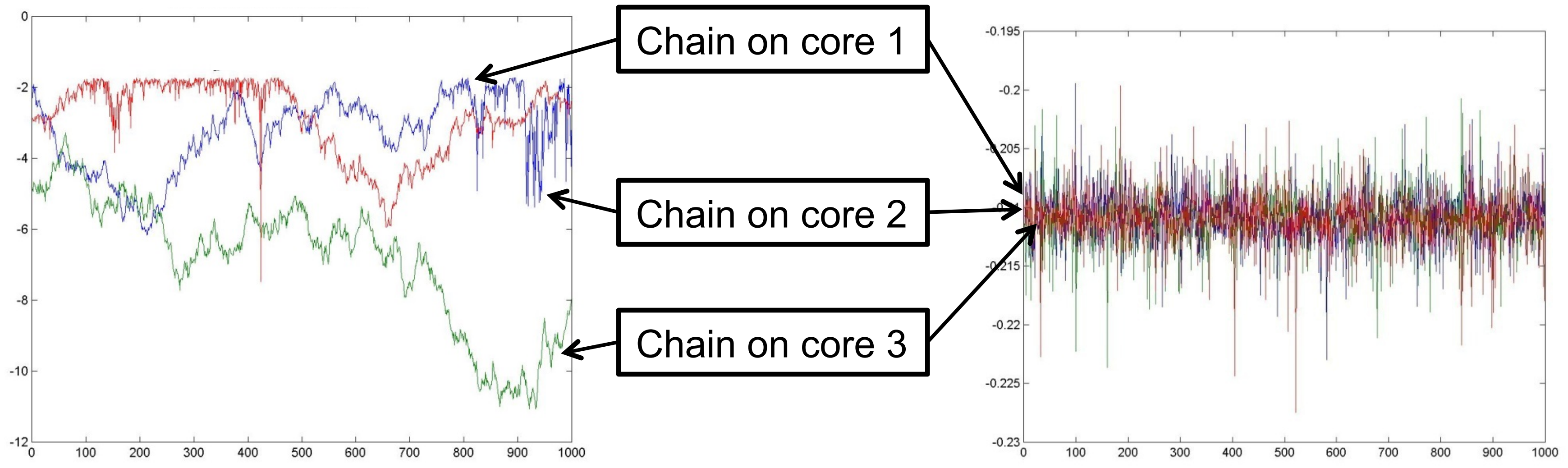
Parallel MCMC for Large Scales

- Datasets and models can be very large
  - Millions to billions of data points
  - Millions to billions of random variables
  - Compute time measured in CPU-*years*
  - Need GBs to TBs of memory
  - E.x. Yahoo web graph has ~1.4 billion nodes and 6.6 billion edges
    - Imagine doing a Markov Random Field on that network

- Without parallelism, we cannot use large datasets and models!
  - Today: how to use multiple CPUs and machines in MCMC

Taking Multiple Chains

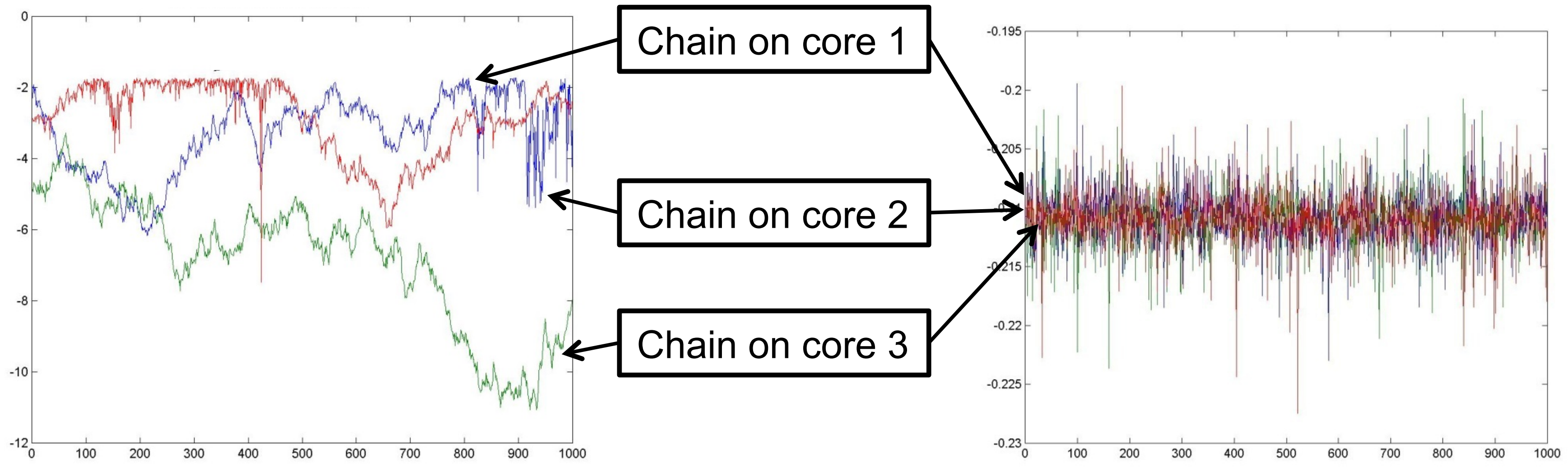
- Proper use of MCMC actually requires parallelism
  - To determine convergence, you need to take multiple MCMC chains
  - Chains are independent, so you can run one chain per CPU
  - Once converged, you can combine samples from all chains



Not converged **Converged**

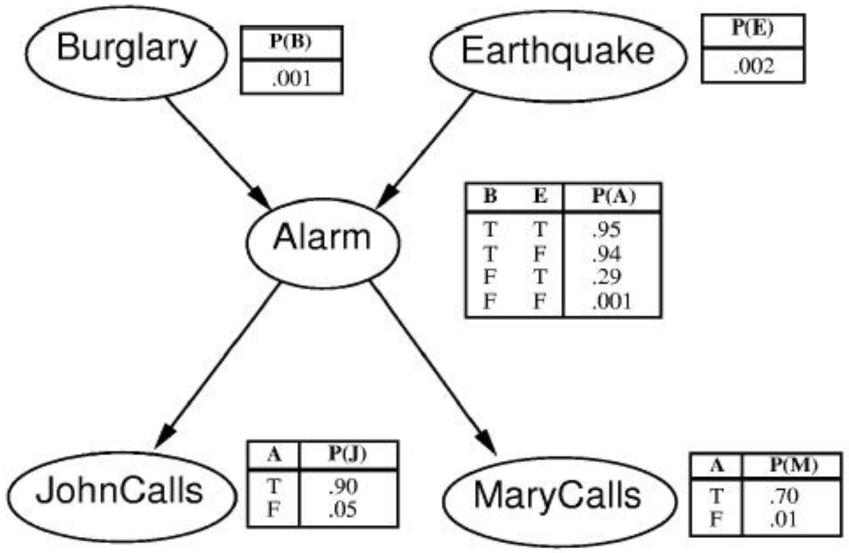
Taking Multiple Chains

- Taking multiple chains doesn't solve all issues, though
  - If burn-in is long, then all chains will take a long time to converge!
  - We need a way to take each sample faster…



Not converged **Converged**

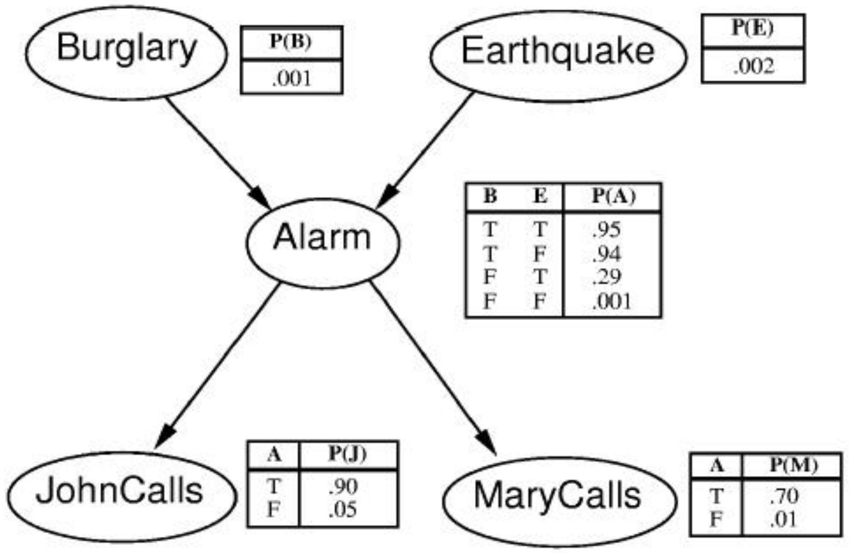
Idea: Run Gibbs Sampling in Parallel?



t	B	E	A	J	M
0	F	F	F	F	F
1					
2					
3					
4					

- Recall the alarm network
  - Initialize all variables at t = 0 to False
  - Idea: parallel Gibbs sample all variables at step t conditioned on t-1

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F				
2					
3					
4					

- Sampling P(B|A,E) at t = 1: Using Bayes Rule,

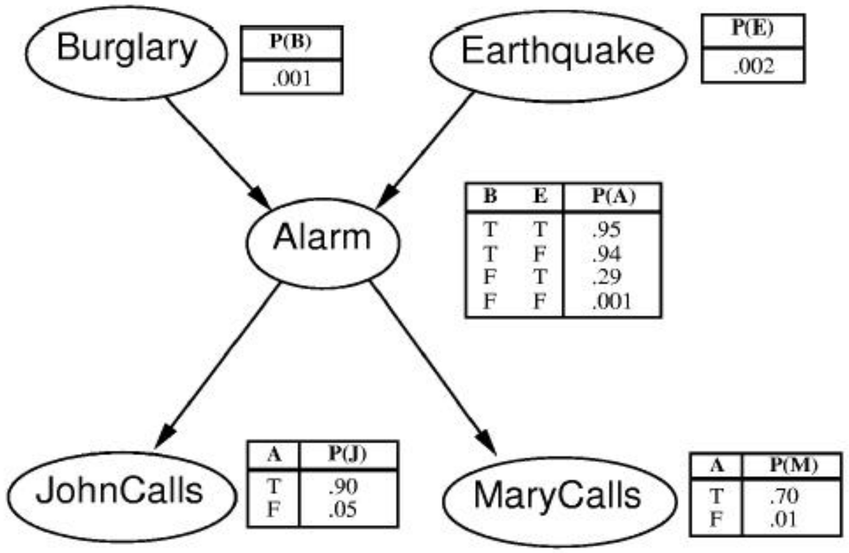
$$P(B \mid A, E) \propto P(A \mid B, E) P(B)$$

- (A,E) = (F,F), so we compute the following, and sample B = F

$$P(B = T \mid A = F, E = F) \propto (0.06)(0.01) = 0.0006$$

$$P(B = F \mid A = F, E = F) \propto (0.999)(0.999) = 0.9980$$

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T			
2					
3					
4					

- Sampling P(E|A,B): Using Bayes Rule,

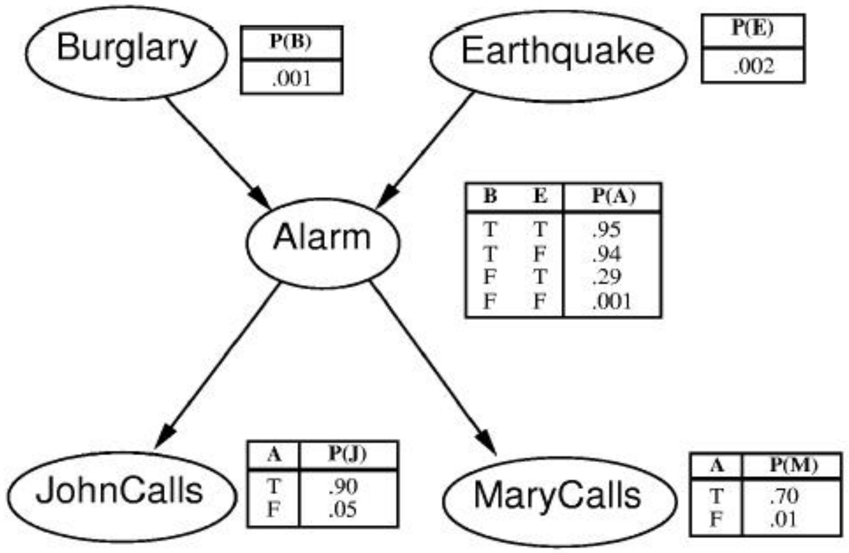
$$P(E \mid A, B) \propto P(A \mid B, E) P(E)$$

- (A,B) = (F,F), so we compute the following, and sample E = T

$$P(E = T \mid A = F, B = F) \propto (0.71)(0.02) = 0.0142$$

$$P(E = F \mid A = F, B = F) \propto (0.999)(0.998) = 0.9970$$

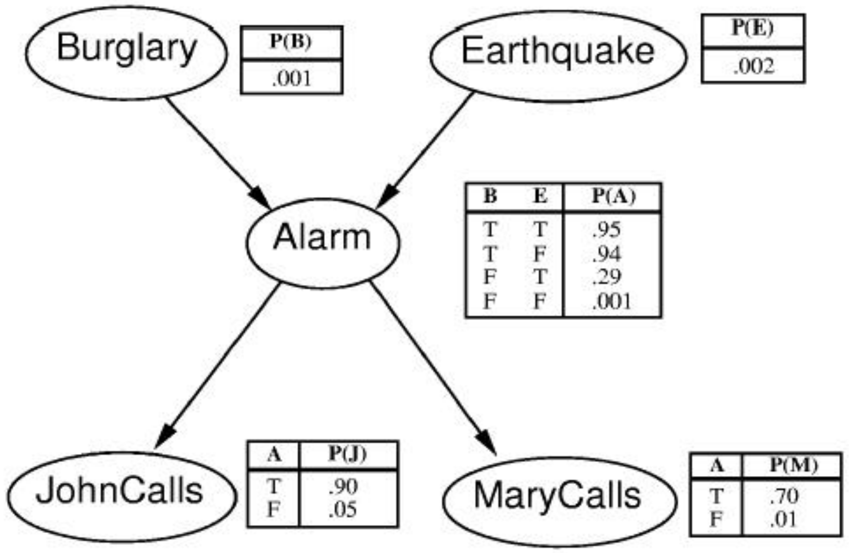
Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T			
2					
3					
4					

- Notice the difference
  - Normal Gibbs sampling: compute P(E|A,B) based on $B_{t=1}$, $A_{t=0}$
  - Naïve Parallel GS: compute P(E|A,B) based on $B_{t=0}$, $A_{t=0}$
  - At step t, always condition on t-1 instead of most recently sampled value

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F		
2					
3					
4					

- Sampling P(A|B,E,J,M): Using Bayes Rule,

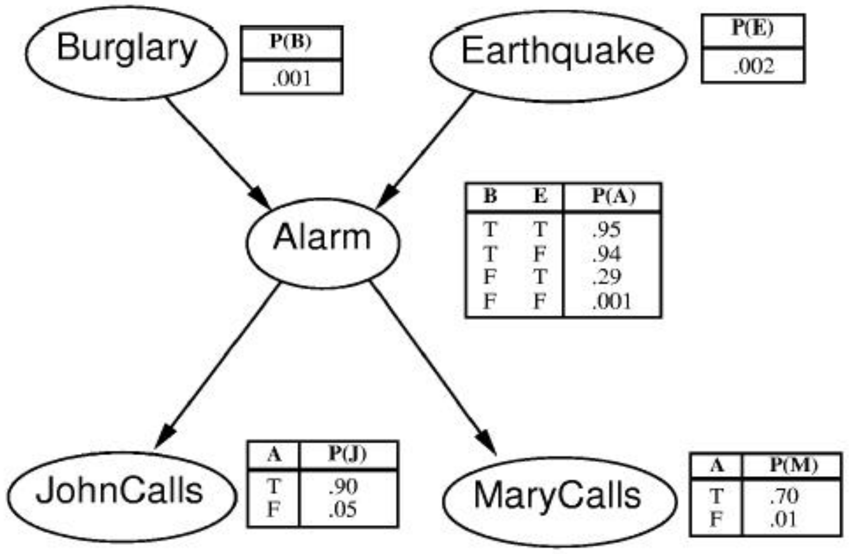
$$P(A \mid B,E,J,M) \propto P(J \mid A)P(M \mid A)P(A \mid B,E)$$

- (B,E,J,M) = (F,F,F,F), so we compute the following, and sample A = F

$$P(A=T \mid B=F,E=F,J=F,M=F) \propto (0.1)(0.3)(0.001) = 0.00003$$

$$P(A=F \mid B=F,E=F,J=F,M=F) \propto (0.95)(0.99)(0.999) = 0.9396$$

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	
2					
3					
4					

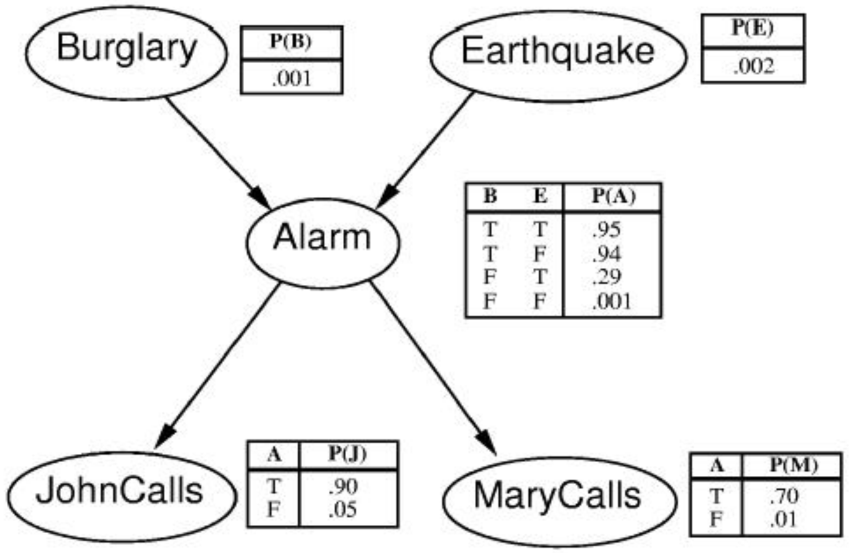
- Sampling P(J|A): No need to apply Bayes Rule

- A = F, so we compute the following, and sample J = T

$$P(J = T \mid A = F) \propto 0.05$$

$$P(J = F \mid A = F) \propto 0.95$$

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	F
2					
3					
4					

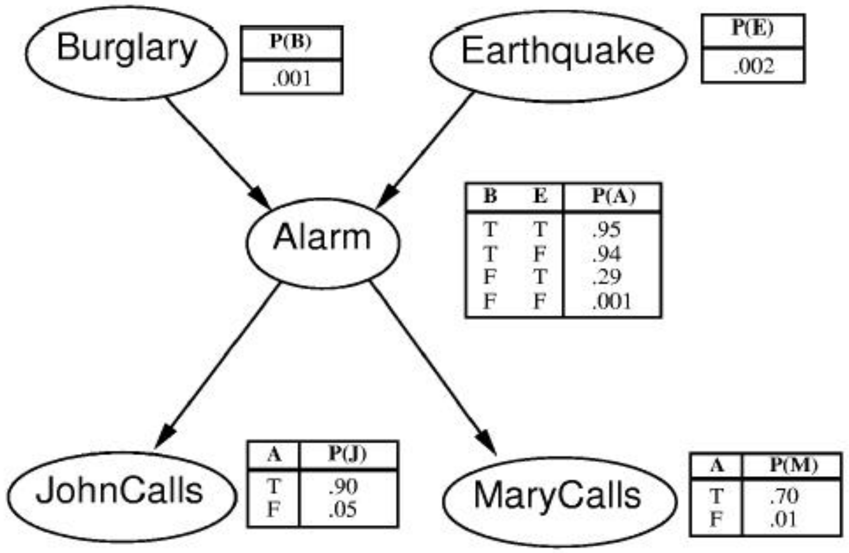
- Sampling P(M|A): No need to apply Bayes Rule

- A = F, so we compute the following, and sample M = F

$$P(M = T \mid A = F) \propto 0.01$$

$$P(M = F \mid A = F) \propto 0.99$$

Naïve Parallel Gibbs Sampling



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	F
2					
3					
4					

- We just finished sampling variables t=1
- Why is the sampling parallelizable?
  - We only conditioned on variable state at t=0, which is known in advance!
  - We can sample B,E,A,J,M on separate processors, without having to send information between processors

Naïve Parallel Gibbs Sampling

- In practice, works very well for some graphical models
  - E.g. collapsed Gibbs Sampling for LDA

$$P(z_i = j | \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta} \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + T\alpha}$$

  - Just assign different *z*$_i$'s to different processors or machines

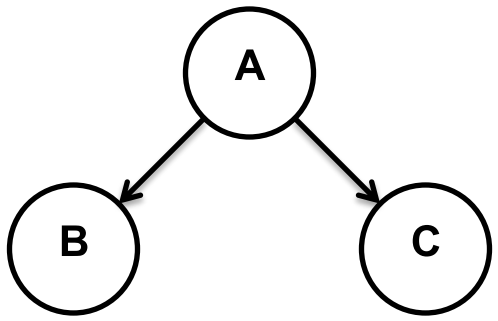
- But there's a problem…

Where Naïve Parallel GS Fails

- Naïve Parallel GS may not converge to the stationary distribution

- Consider the following Bayes Net:

A	P(B)
T	0.0001
F	0.9999

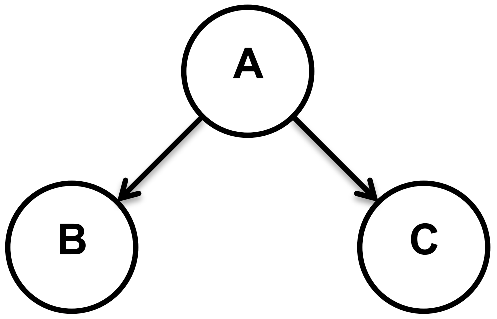


A	P(C)
T	0.0001
F	0.9999

  - Essentially an XOR relation between (A,B) and (A,C)
  - Joint distribution P(A,B,C) has only 8 states, so we can compute the stationary distribution. It is dominated by 2 equally-probable states:
    - (A,B,C) = (T,F,T) and (A,B,C) = (F,T,F)

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



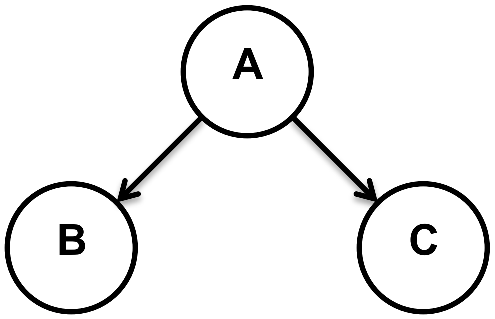
A	P(C)
T	0.0001
F	0.9999

t	A	B	C
0	F	F	F
1			
2			
3			
4			

- Let's initialize (A,B,C) = (F,F,F) and see what happens when we naively Gibbs sample in parallel…

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



A	P(C)
T	0.0001
F	0.9999

t	A	B	C
0	F	F	F
1	T		
2			
3			
4			

- Sampling P(A|B,C):

$$P(A \mid B, C) \propto P(B \mid A) P(C \mid A)$$

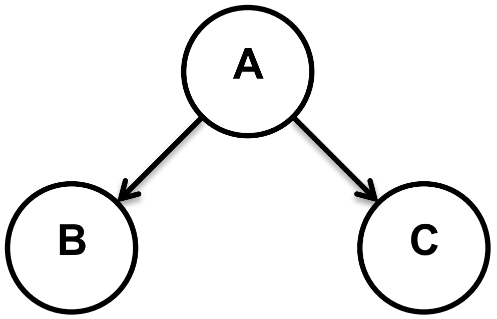
- (B,C) = (F,F) so we sample A = T

$$P(A = T \mid B = F, C = F) \propto (0.999)(0.999) \approx 1$$

$$P(A = F \mid B = F, C = F) \propto (0.001)(0.001) \approx 0$$

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



A	P(C)
T	0.0001
F	0.9999

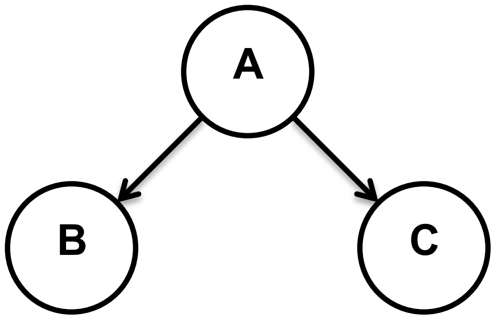
t	A	B	C
0	F	F	F
1	T	T	
2			
3			
4			

- Sampling P(B|A): No need to apply Bayes Rule
- A = F so we sample B = T

$$P(B=T \mid A=F) \propto (0.999) \approx 1$$
$$P(B=F \mid A=F) \propto (0.001) \approx 0$$

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



A	P(C)
T	0.0001
F	0.9999

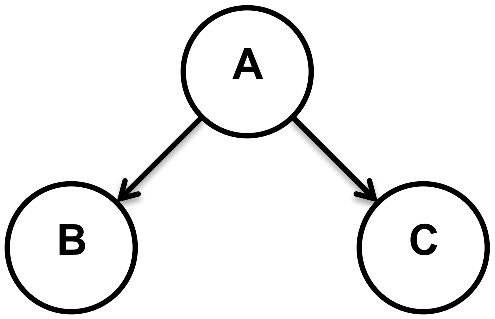
t	A	B	C
0	F	F	F
1	T	T	T
2			
3			
4			

- Sampling P(C|A): No need to apply Bayes Rule
- A = F so we sample C = T

$$P(C = T \mid A = F) \propto (0.999) \approx 1$$
$$P(C = F \mid A = F) \propto (0.001) \approx 0$$

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



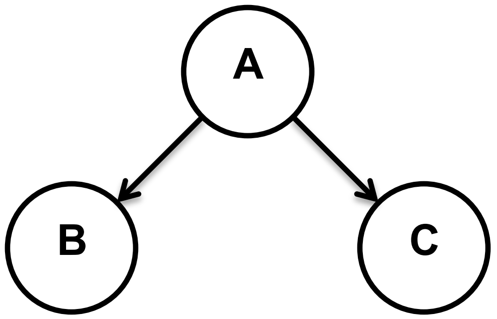
A	P(C)
T	0.0001
F	0.9999

t	A	B	C
0	F	F	F
1	T	T	T
2	F	F	F
3			
4			

- Easy to see that at t=2, we will get (A,B,C) = (F,F,F)

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



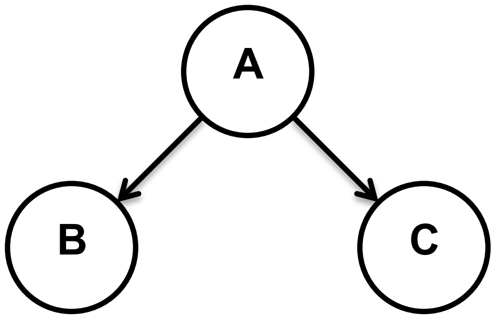
A	P(C)
T	0.0001
F	0.9999

t	A	B	C
0	F	F	F
1	T	T	T
2	F	F	F
3	T	T	T
4			

- Easy to see that at t=2, we will get (A,B,C) = (F,F,F)
- At t=3, (A,B,C) = (T,T,T)

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



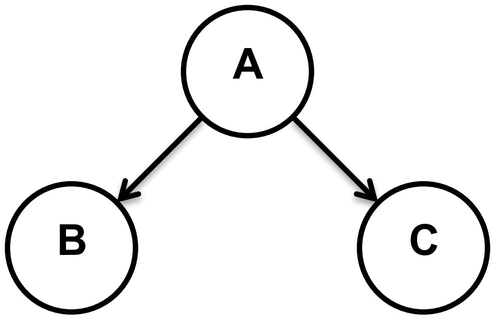
A	P(C)
T	0.0001
F	0.9999

t	A	B	C
0	F	F	F
1	T	T	T
2	F	F	F
3	T	T	T
4	F	F	F

- Easy to see that at t=2, we will get (A,B,C) = (F,F,F)
- At t=3, (A,B,C) = (T,T,T)
- At t=4, (A,B,C) = (F,F,F)

Where Naïve Parallel GS Fails

A	P(B)
T	0.0001
F	0.9999



A	P(C)
T	0.0001
F	0.9999

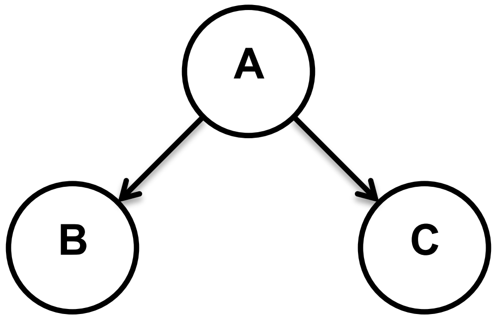
t	A	B	C
0	F	F	F
1	T	T	T
2	F	F	F
3	T	T	T
4	F	F	F

- Easy to see that at t=2, we will get (A,B,C) = (F,F,F)
- At t=3, (A,B,C) = (T,T,T)
- At t=4, (A,B,C) = (F,F,F)
- Can you see the problem?

Where Naïve Parallel GS Fails

- We know the stationary distribution is [(F,T,F), (T,F,T)]
  - But naïve parallel GS gets stuck in [(T,T,T), (F,F,F)]

A	P(B)
T	0.0001
F	0.9999



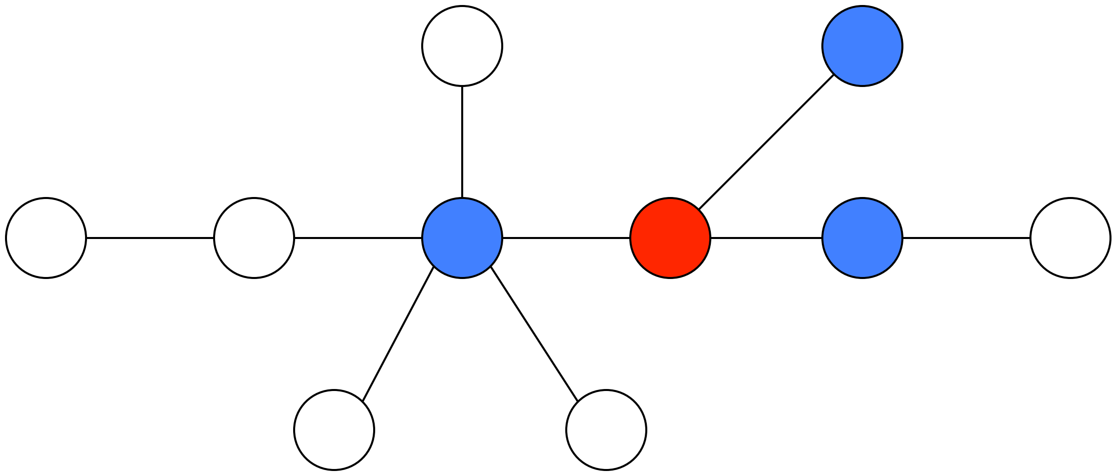
A	P(C)
T	0.0001
F	0.9999

  - Naïve parallel GS performs poorly on near-discrete distributions

- What is the correct way to Gibbs sample in parallel?

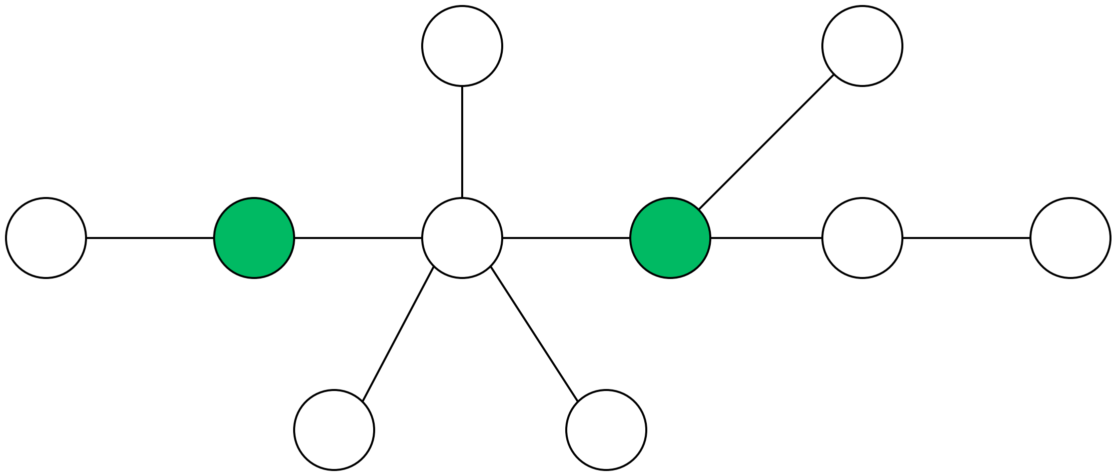
Correct Parallel Gibbs Sampling

- Recall that in MRFs, we Gibbs sample by sampling from P(x| MB(x)), the conditional distribution of x given its Markov Blanket MB(x)
  - For MRFs, the Markov Blanket of x is just its neighbors
  - In the MRF below, the red node's Markov Blanket consists of the blue nodes



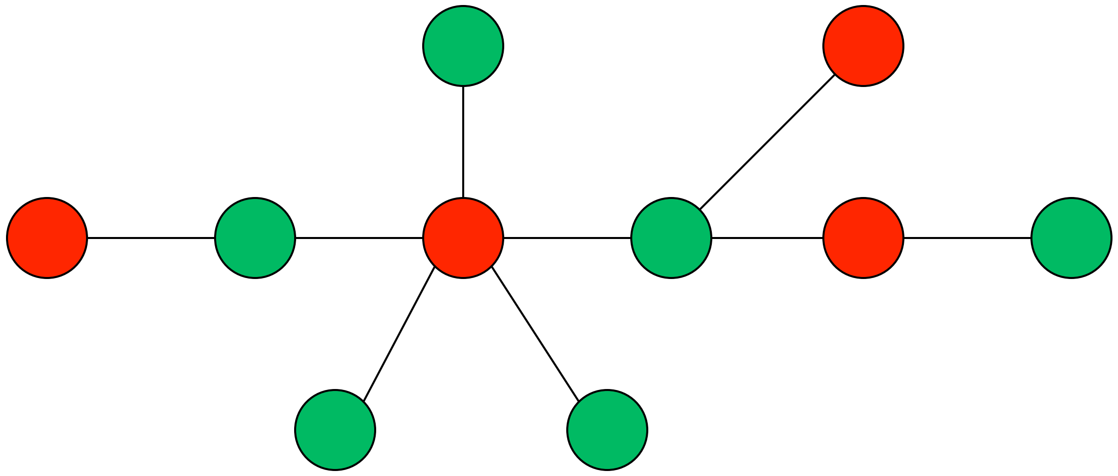
Correct Parallel Gibbs Sampling

- Observe that we can *correctly* Gibbs sample the two green nodes simultaneously
  - Neither node is part of the other's Markov Blanket, so their conditional distributions do not depend on each other
  - Sampling one of the green nodes doesn't change the conditional distribution of the other node!



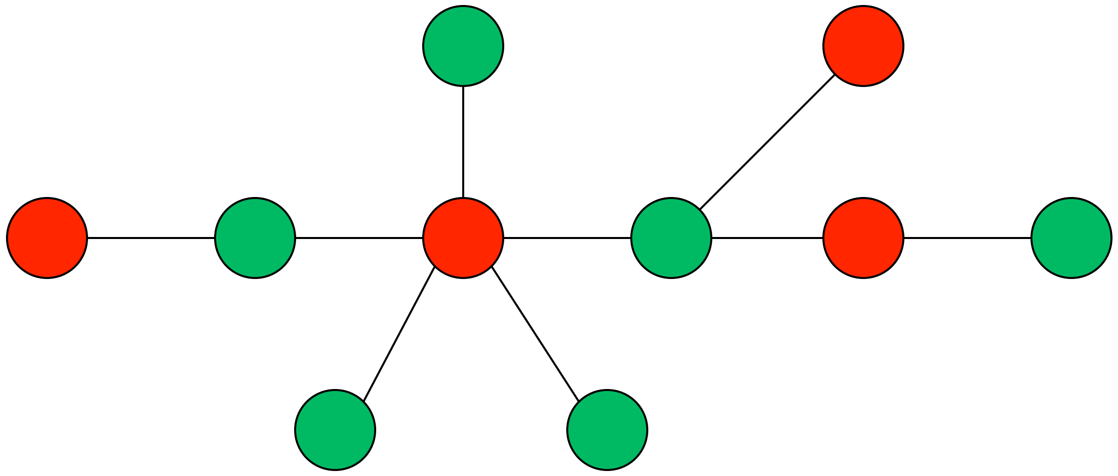
Correct Parallel Gibbs Sampling

- How do we generalize this idea to the whole graph?
  - Find subsets of nodes, such that all nodes in a given subset are not in each other's Markov Blankets, and the subsets cover the whole graph
    - The subsets should be as large as possible
      - Because we can Gibbs sample all nodes in a subset at the same time
    - At the same time, we want as few subsets as possible
      - The Markov Blankets of different subsets overlap, so they cannot be sampled at the same time. We must process the subsets sequentially.



Correct Parallel Gibbs Sampling

- We can find these covering subsets with k-coloring algorithms (Gonzales et al., 2011)
  - A k-coloring algorithm colors a graph using k colors, such that:
    - Every node gets one color
    - No edge has two nodes of the same color
- Trees always admit a 2-coloring (e.g. below)
  - Assign one color to some node, and alternate colors as you move away



Correct Parallel Gibbs Sampling

- Bipartite graphs are always 2-colorable
  - Color each side of the bipartite graph with opposite colors
  - e.x. Latent Dirichlet Allocation model is bipartite

- However, not all graphs have k-colorings for all $k \geq 2$
  - In the worst case, a graph with n nodes can require n colors
    - The full clique is one such graph
  - Determining if a graph is k-colorable for $k > 2$ is NP-complete
  - In practice, we employ heuristics to find k-colorings

- Instead of using k-colorings, why not just Gibbs sample all variables at the same time?
  - The Markov Chain may become non-ergodic, and is no longer guaranteed to converge to the stationary distribution!

Online Parallel MCMC

- In "online" algorithms, we need to process new data points one-at-a-time
  - Moreover, we have to "forget" older data points because memory is finite
- For such applications to be viable, we can only afford constant time work per new data point
  - Otherwise we will reach a point where new data can no longer be processed in a reasonable amount of time
- We also want the algorithm to be parallel for scaling up
- What MCMC techniques can we use to make an online parallel algorithm?

Sequential Monte Carlo

- SMC is a generalization of Particle Filters
  - Recall that PFs incrementally sample $P(X_t|Y_{1:t})$, where the Xs are latent r.v.s and the Ys are observations under a state-space model
  - SMC does not assume the GM is a state-space model, or has any particular structure at all

- Suppose we have n r.v.s $x_1,\dots,x_n$
  - SMC first draws samples from the marginal distribution $P(x_1)$, then $P(x_{1:2})$, and so on until $P(x_{1:n})$
  - Key idea: Construct proposals such that we sample from $P(x_{1:k+1})$ in constant time, given samples from $P(x_{1:k})$
  - Like other MCMC algorithms, we only require that we can evaluate $P'(x_{1:n}) = aP(x_{1:n})$ for some unknown a

Sequential Importance Sampling

- SIS is the foundation of Sequential Monte Carlo
  - It allows new variables to be sampled in constant time, without resampling older variables
- SIS uses proposal distributions with the following structure:

$$\begin{aligned} q_n(x_{1:n}) &= q_{n-1}(x_{1:n-1}) q_n(x_n \mid x_{1:n-1}) \\ &= q_1(x_1) \prod_{k=2}^{n} q_k(x_k \mid x_{1:k-1}) \end{aligned}$$

  - Notice we can propose x_{k+1} if we've already drawn $x_{1:k}$, without having to redraw $x_{1:k}$

Sequential Importance Sampling

- In normalized importance sampling, recall how the sample weights w^i are defined:

$$\langle f(X) \rangle_P = \sum_i f(x^i) w^i$$

where $w^i = \frac{r^i}{\sum_j r^j}$ and $r^i = \frac{P'(x^i)}{Q(x^i)}$

- In SIS, the unnormalized weights r can be rewritten as a telescoping product:

$$\begin{aligned} r(x_{1:n}) &= \frac{P'_n(x_{1:n})}{q_n(x_{1:n})} \\ &= \frac{P'_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{P'_n(x_{1:n})}{P'_{n-1}(x_{1:n-1}) q_n(x_n \mid x_{1:n-1})} \\ &= r_{n-1}(x_{1:n-1}) \alpha_n(x_{1:n}) \\ &= r_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}) \end{aligned}$$

where $\alpha_n(x_{1:n}) = \frac{P'_n(x_{1:n})}{P'_{n-1}(x_{1:n-1}) q_n(x_n \mid x_{1:n-1})}$

Sequential Importance Sampling

$$r(x_{1:n}) = r_1(x_1)\prod_{k=2}^{n}\alpha_k(x_{1:k}) \qquad \text{where} \qquad \alpha_n(x_{1:n}) = \frac{P'_n(x_{1:n})}{P'_{n-1}(x_{1:n-1})q_n(x_n \mid x_{1:n-1})}$$

- This means the unnormalized weights r can be computed incrementally
  - Compute α_n and use it to update $r(x_{1:n-1})$ to $r(x_{1:n})$
    - NB: For this update to be constant time, we also require $P'_n(x_{1:n})$ to be computable from $P'_{n-1}(x_{1:n-1})$ in constant time
  - We remember the unnormalized weights r at each iteration, and compute the normalized weights w as needed from r
- Thus, we can sample x AND compute the normalized weights w using constant time per new variable x_n
  - So SIS meets the requirements for an online inference algorithm!
- Even better, the samples don't depend on each other
  - Assign one CPU core per sample to make the SIS algorithm parallel!

Sequential Importance Sampling

- SIS algorithm:
  - At time n = 1
    - Parallel draw samples $x^i_1 \sim q_1(x_1)$
    - Parallel compute unnormalized weights $r^i_1 = P'_1(x^i_1) / q_1(x^i_1)$
    - Compute normalized weights w^i_1 by normalizing r^i_1
      - Although this step is sequential, it takes almost no time to perform
  - At time n ≥ 2
    - Parallel draw samples $x^i_n \sim q_n(x_n|x^i_{1:n-1})$
    - Parallel compute unnorm. wgts. $r^i_n = r^i_{n-1}\alpha_n(x^i_{1:n}) = r^i_{n-1}\dfrac{P'_n(x^i_{1:n})}{P'_{n-1}(x^i_{1:n-1})q_n(x^i_n \mid x^i_{1:n-1})}$
    - Compute normalized weights w^i_n by normalizing r^i_n
      - Although this step is sequential, it takes almost no time to perform

Sequential Importance Sampling

- But we are not done yet!

- Unfortunately, SIS suffers from a severe drawback: the variance of the samples increases exponentially with n!
  - See eq (31) of Doucet’s SMC tutorial for an example

- Resampling at each iteration will decrease the sample variance!
  - Similar to weighted resampling from the first MC lecture!

Multinomial Resampling

- Suppose we have m samples $x^1,\ldots,x^m$ with corresponding importance weights $w^1,\ldots,w^m$

- Construct a categorical distribution from these samples:
  - This distribution has m categories (choices)
  - The probability of drawing category k is w^k
  - Drawing category k gets us x^k

- To resample, just draw N times from this distribution
  - Note that N can be greater/less than m!

- For more advanced strategies such as systematic and residual resampling, refer to page 13 of Doucet's SMC tutorial

Why Resample?

- Apart from decreasing variance, there are other reasons…

- Resampling removes samples x^k with low weights w^k
  - Low-weight samples come from low-probability regions of P(x)
    - We want to focus computation on high-probability regions of P(x)
  - Notice that each sample gets an equal amount of computation, regardless of its weight w_k
    - Resampling ensures that more computation is spent on samples x_k that come from high-probability regions of P(x)

- Resampling prevents a small number of samples x_k from dominating the empirical distribution
  - Resampling resets all weights w_k to 1/N
    - This prevents sample weights w_k from growing until they reach 1

Sequential Monte Carlo

- The SMC algorithm is just SIS with resampling:
  - At time n = 1
    - Parallel draw samples $x^i_1 \sim q_1(x_1)$
    - Parallel compute unnormalized weights $r^i_1 = P'_1(x^i_1) / q_1(x^i_1)$
    - Compute normalized weights w^i_1 by normalizing r^i_1
    - Parallel resample w^i_1, x^i_1 into N equally-weighted particles x^i_1
  - At time n ≥ 2
    - Parallel draw samples $x^i_n \sim q_n(x_n|x^i_{1:n-1})$
    - Parallel compute unnorm. wgts. $r^i_n = r^i_{n-1}\alpha_n(x^i_{1:n}) = r^i_{n-1}\dfrac{P'_n(x^i_{1:n})}{P'_{n-1}(x^i_{1:n-1})q_n(x^i_n \mid x^i_{1:n-1})}$
    - Compute normalized weights w^i_n by normalizing r^i_n
    - Parallel resample w^i_n,$x^i_{1:n}$ into N equally-weighted particles $x^i_{1:n}$

Summary

- Parallel Gibbs sampling
  - Naïve strategy: sample all variables at the same time
  - Correct strategy: perform graph colorings and sample same-colored nodes in parallel

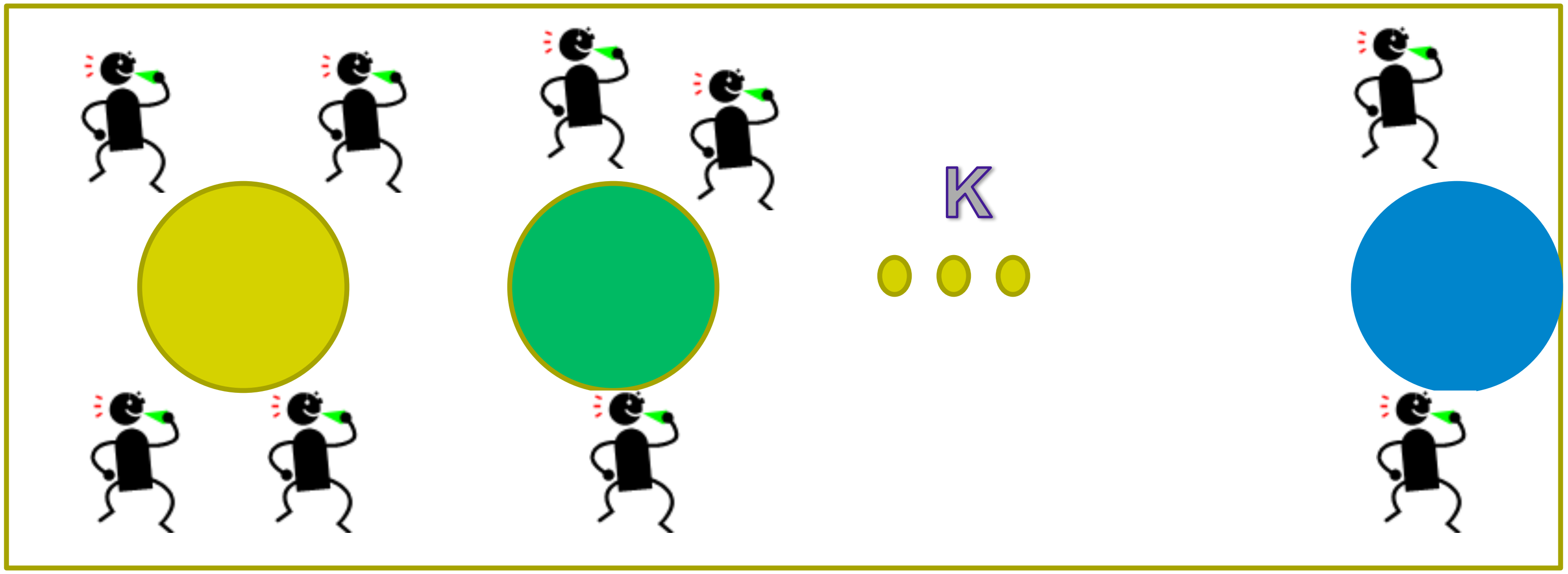
- Sequential Monte Carlo
  - Uses incremental proposal distributions
  - Provides a framework for designing online, parallel MCMC algorithms

Parallel Inference for Bayesian Nonparametric

- Dirichlet Process Mixture Model (recap)
- Inference schemes (recap)
- Parallel inference schemes
- Results

Finite Mixture Model:- Restaurant Perspective

People sit on the table with the most preferred dish/color

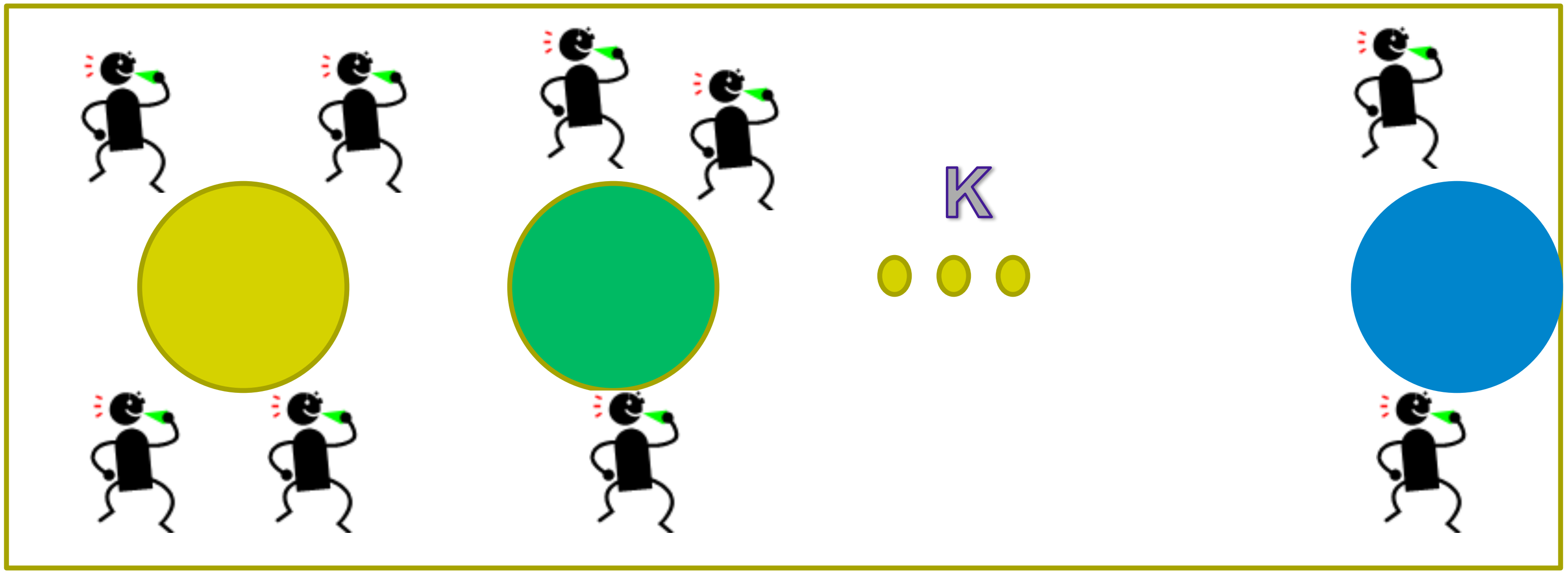


Finite Mixture Model:- Restaurant Perspective

- Table:
  - Cluster
- People:
  - Items to be clustered
- Parameters:
  - Dish/color on each table
    - Center of each cluster
- Hidden Variable:
  - Assignment of people to each table

Finite Mixture Model:- Restaurant Perspective

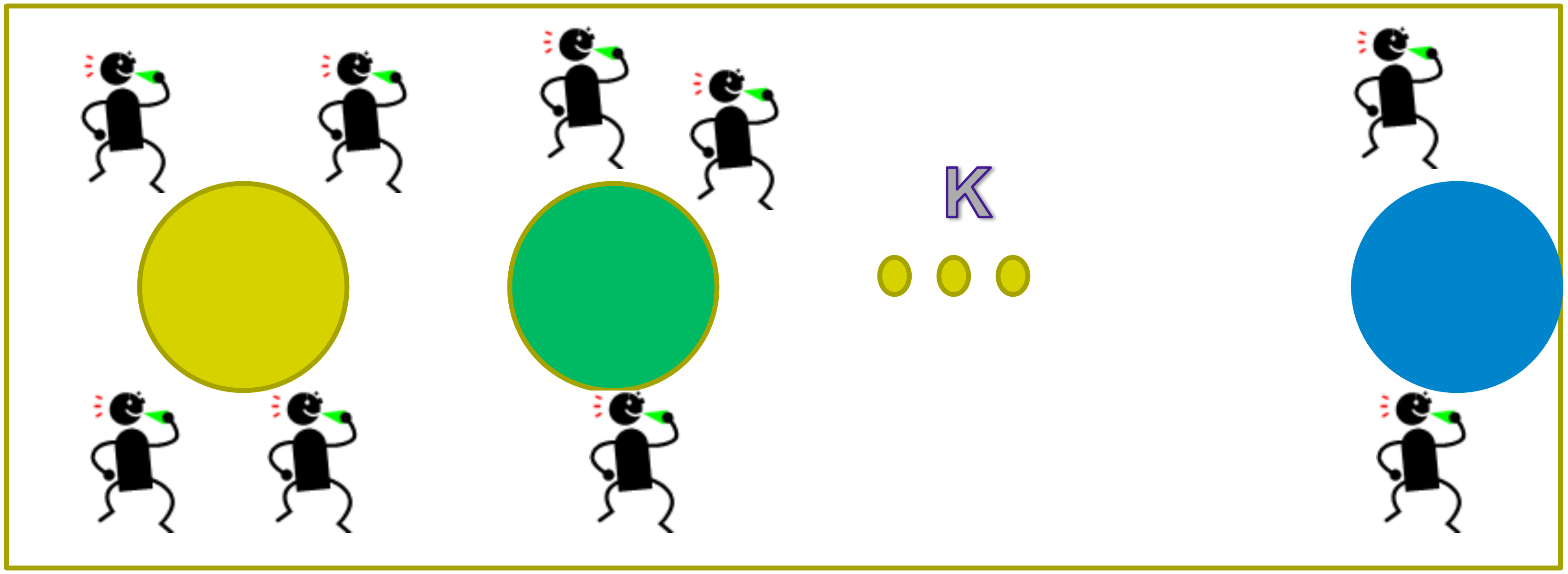
People sit on the table with the most preferred dish/color



Which clustering algorithm will it lead to?

Finite Mixture Model:- Restaurant Perspective

People sit on the table with the most preferred dish/color

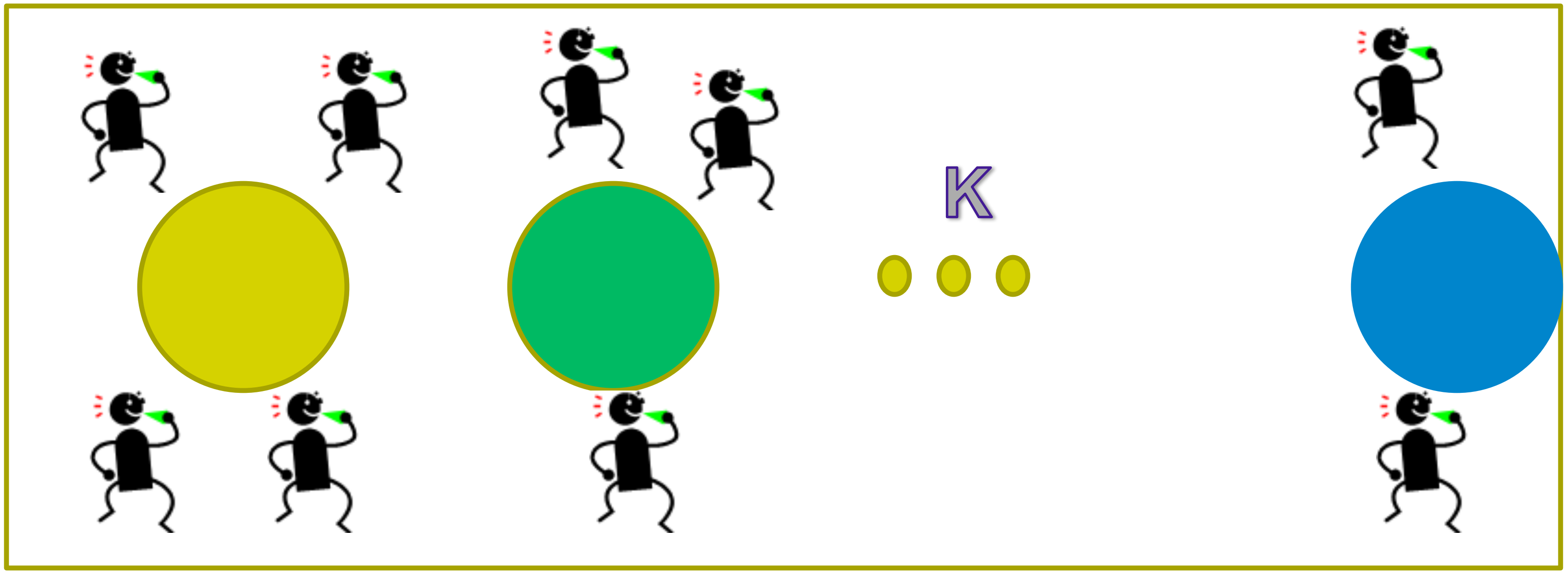


Which clustering algorithm will it lead to?

Hard Kmeans

Finite Mixture Model:- Restaurant Perspective

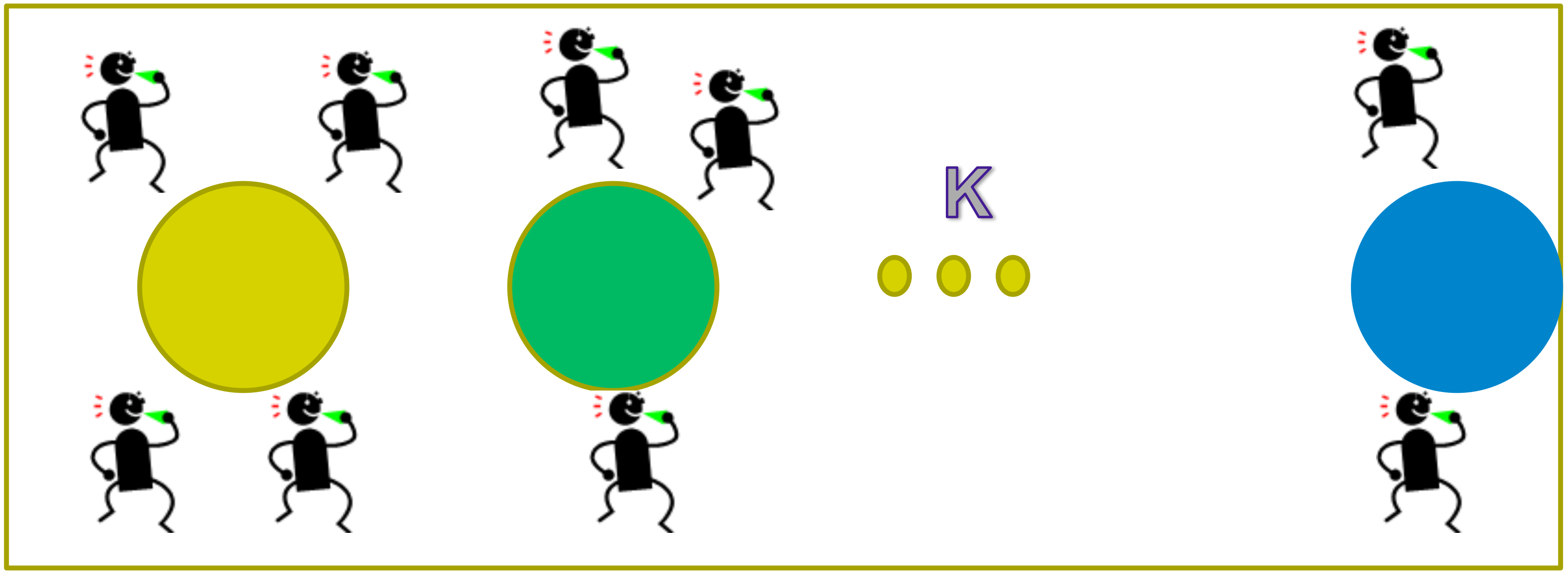
People sit on the table proportional to appreciation of dish/color



Which clustering algorithm will it lead to?

Finite Mixture Model:- Restaurant Perspective

People sit on the table proportional to appreciation of dish/color

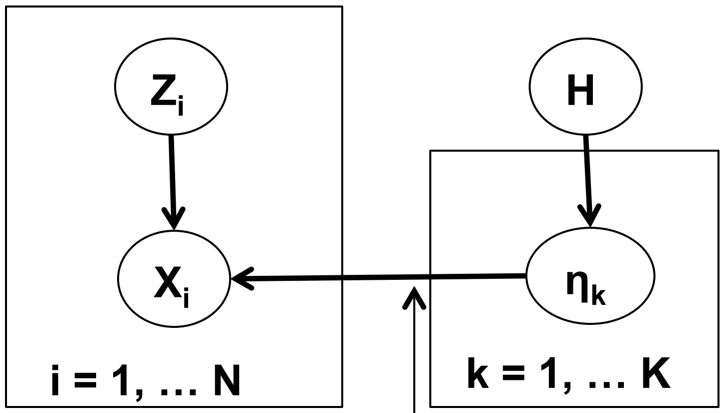


Which clustering algorithm will it lead to?

Soft Kmeans

Soft Kmeans Generative Model

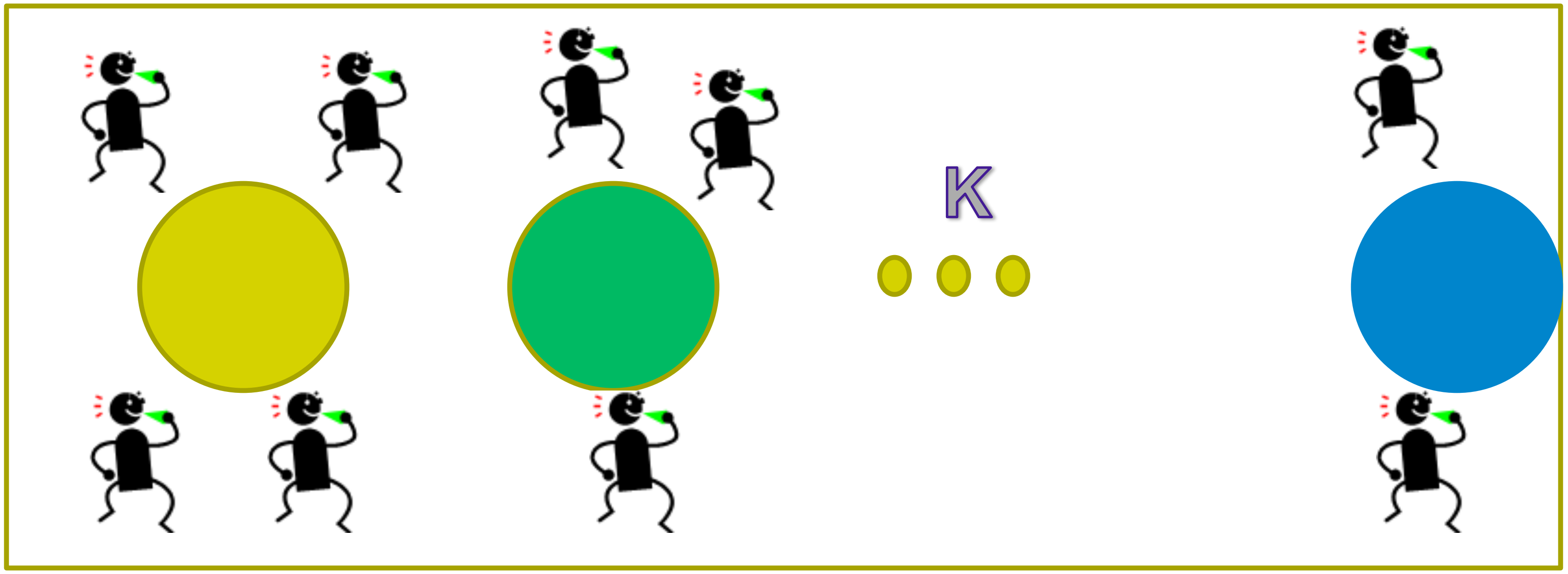
```
for k=1, … K
        $\eta_k \sim H$
for i=1, … N
      $Z_i \sim U(1,K)$
      $X_i \sim f(\eta_{zi})$
```



appreciation of dish/color

Finite Mixture Model:- Restaurant Perspective

People sit on the table proportional to appreciation of dish/color and number of people sitting on the table

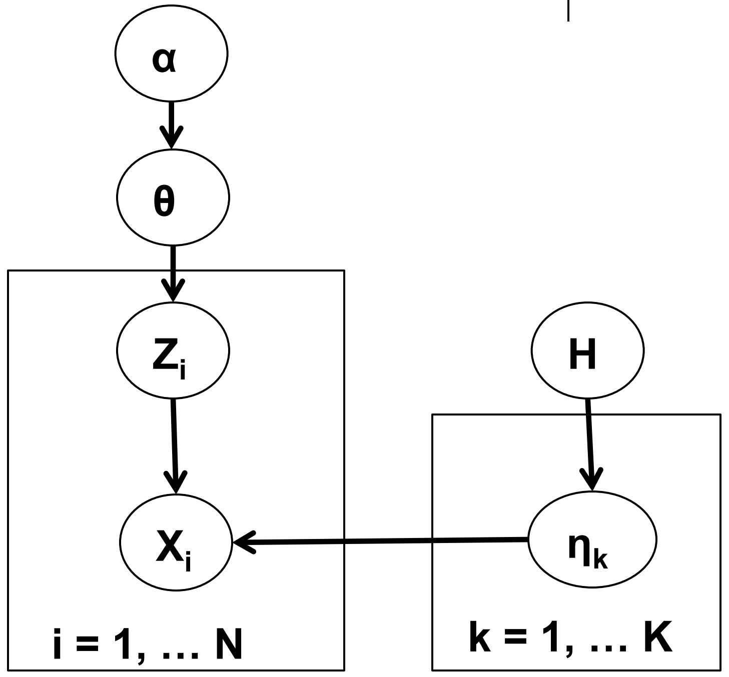


Which clustering algorithm will it lead to?

Dirichlet Distribution Mixture Model

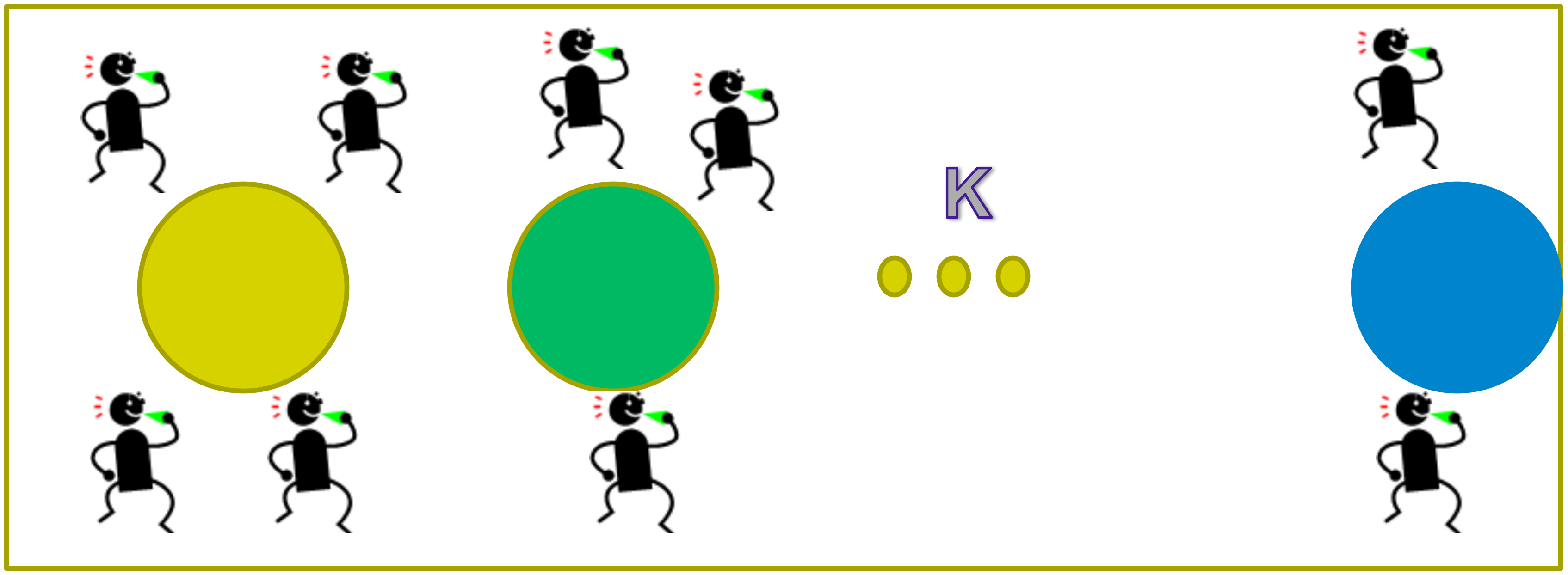
Finite MM Generative Model

```
for k=1, … K
    $\eta_k \sim H$
$\theta \sim Dir(\alpha)$
for i=1, … N
    $Z_i \sim Mul(\theta)$
    $X_i \sim f(\eta_{z_i})$
```



Finite Mixture Model:- Restaurant Perspective

People sit on the table proportional to appreciation of dish/color and number of people sitting on the table

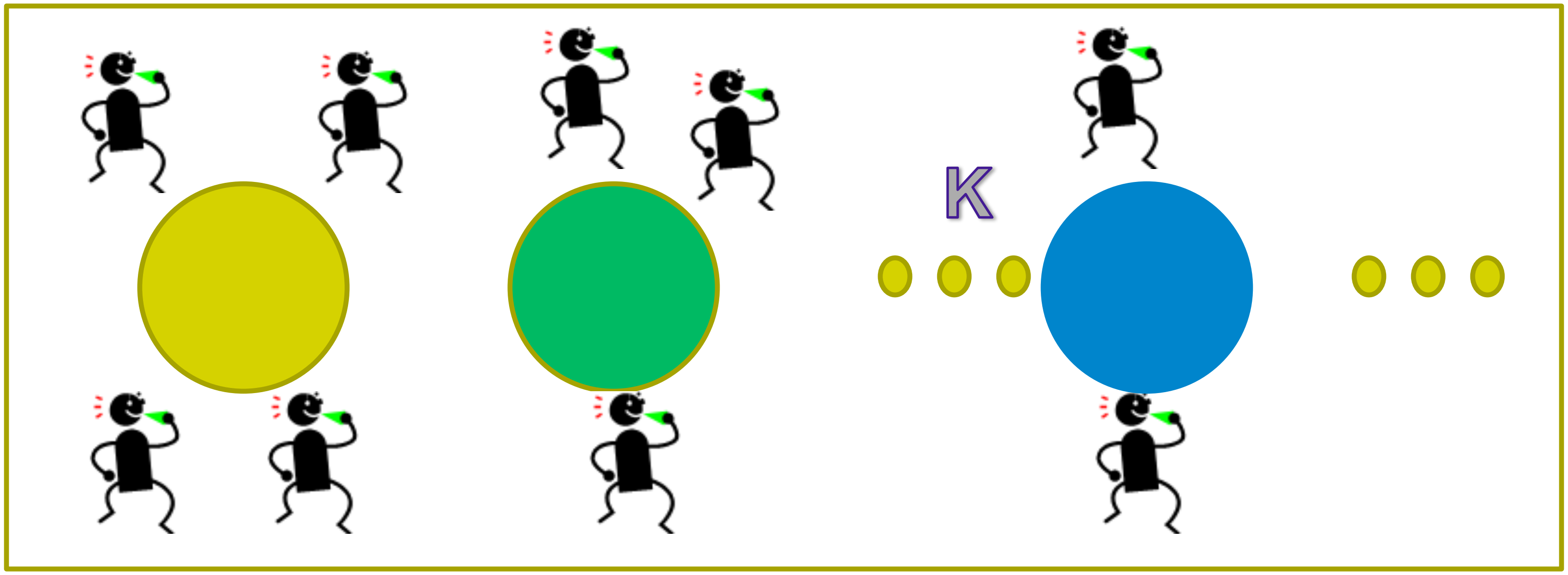


Which clustering algorithm will it lead to?

Dirichlet Distribution
Mixture Model

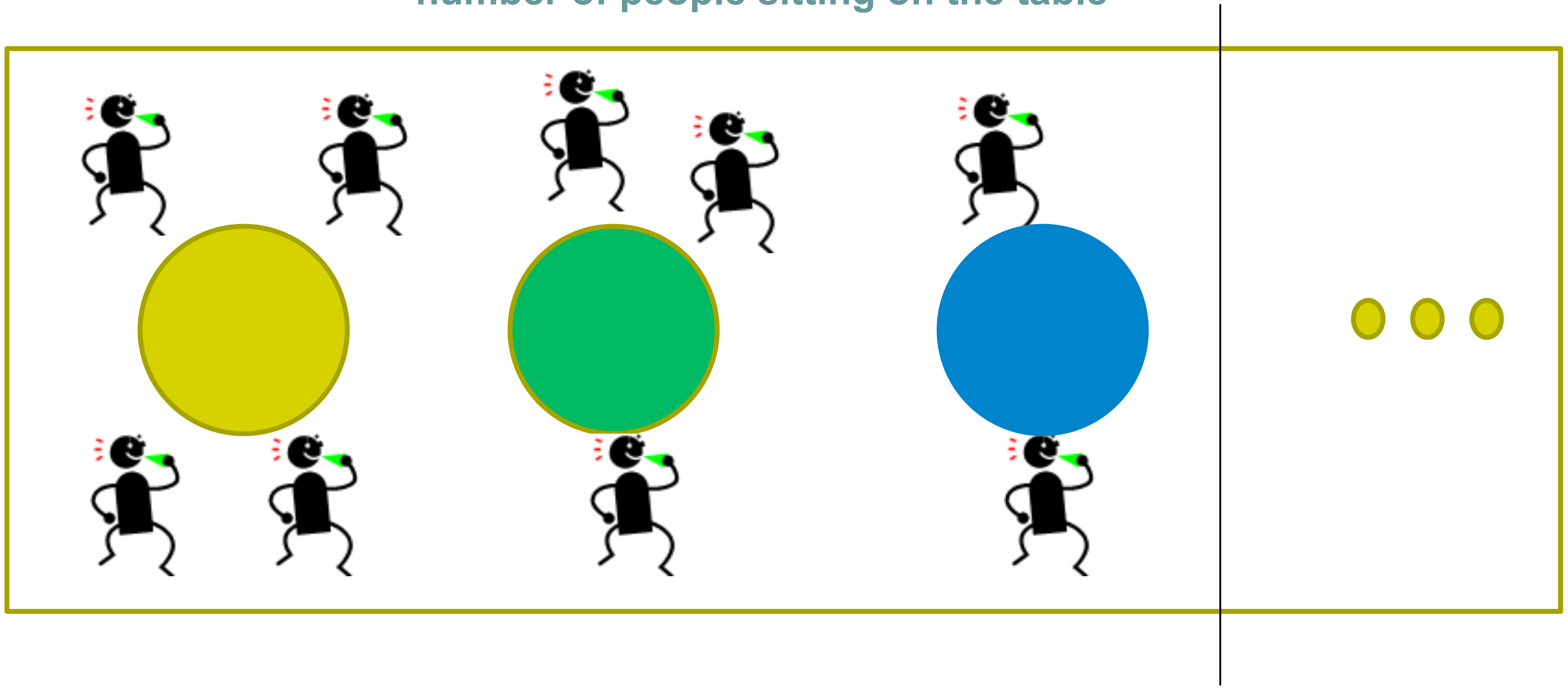
Infinite Mixture Model:- Restaurant Perspective

People sit on the table proportional to appreciation of dish/color and number of people sitting on the table



Infinite Mixture Model:- Restaurant Perspective

People sit on the table proportional to appreciation of dish/color and number of people sitting on the table

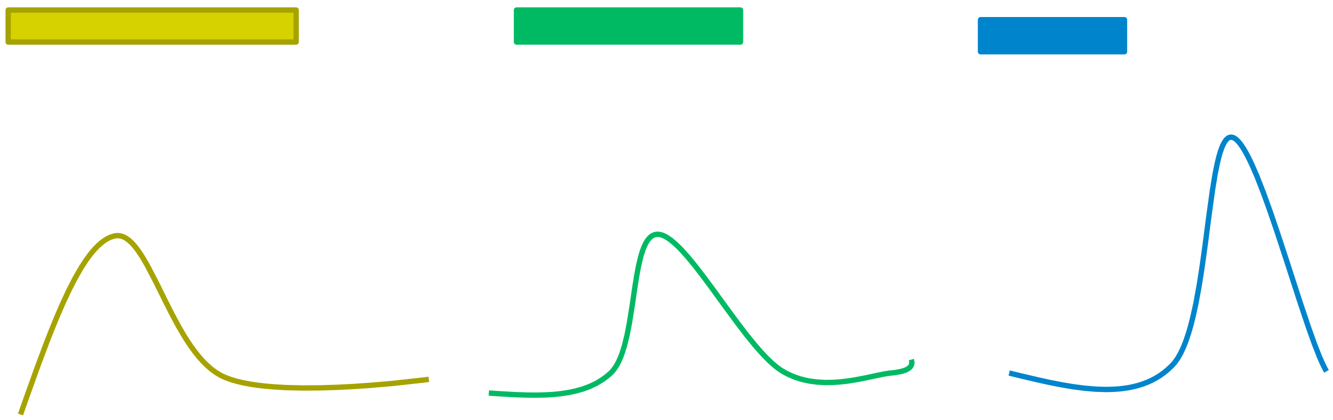


$\frac{4}{9+\alpha}$ $\frac{3}{9+\alpha}$ $\frac{2}{9+\alpha}$ $\frac{\alpha}{9+\alpha}$

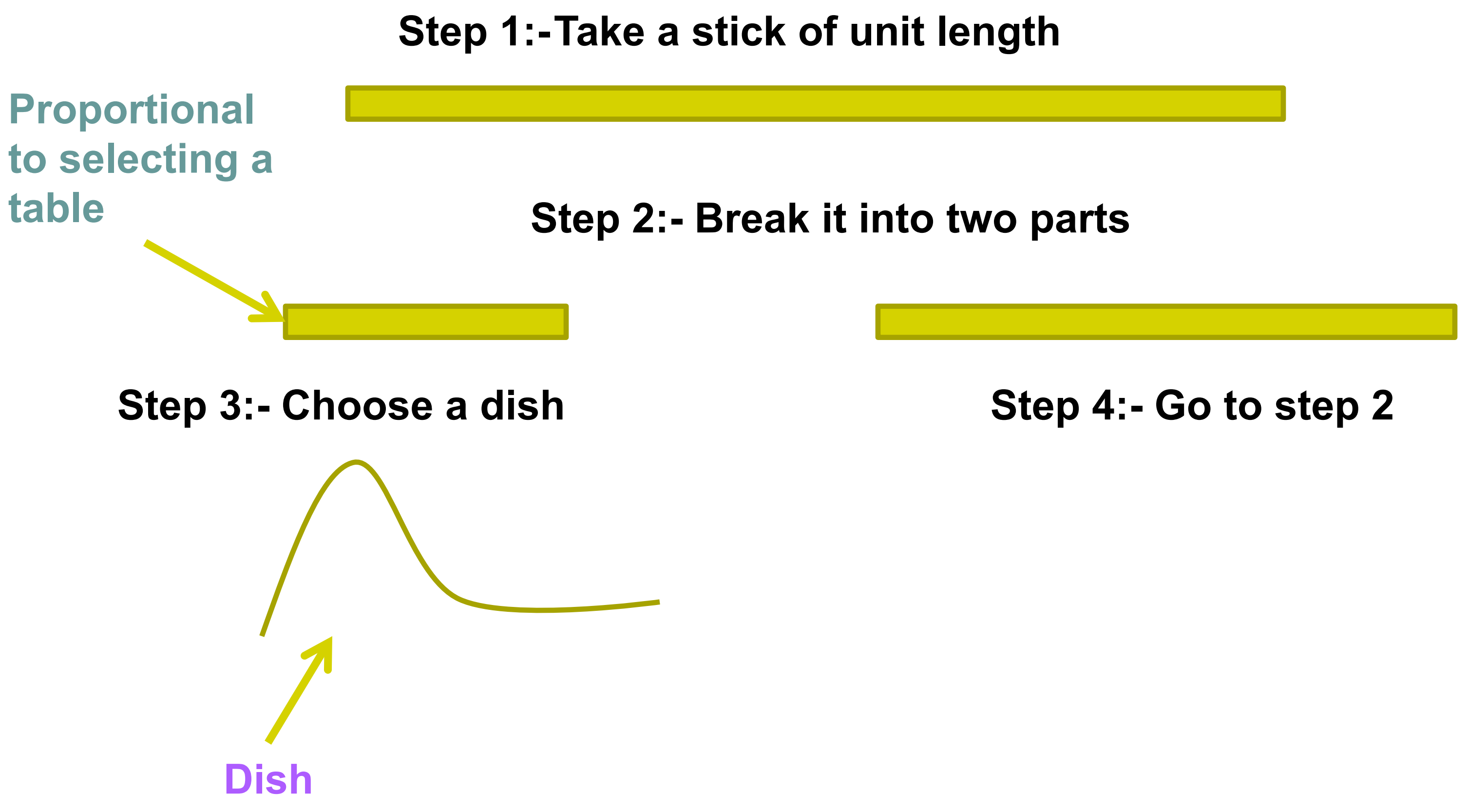
Turning the definition

Proportional to selecting a table

Dish on the table



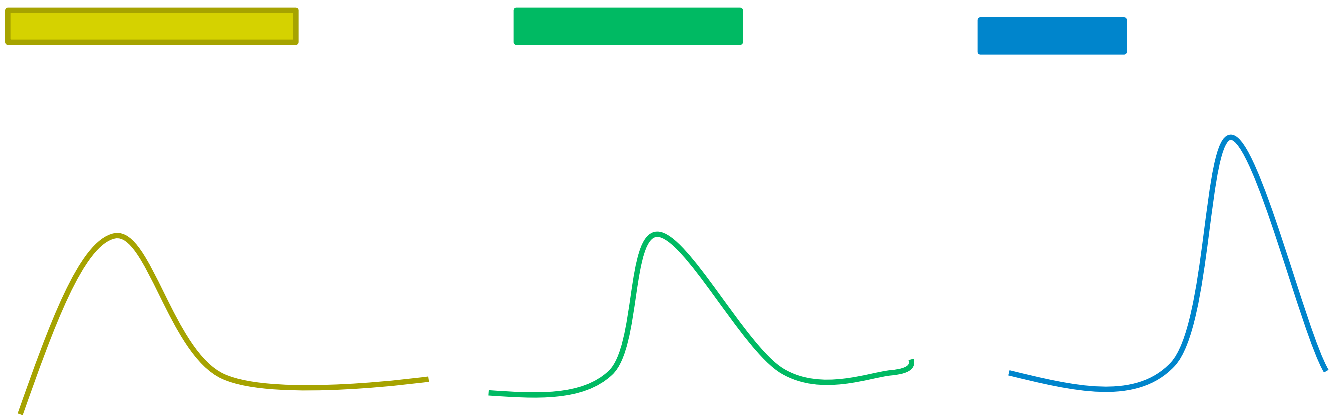
Stick Breaking Construction


Step 1:-Take a stick of unit length
Proportional to selecting a table
Step 2:- Break it into two parts
Step 3:- Choose a dish
Step 4:- Go to step 2
Dish

Stick Breaking Construction

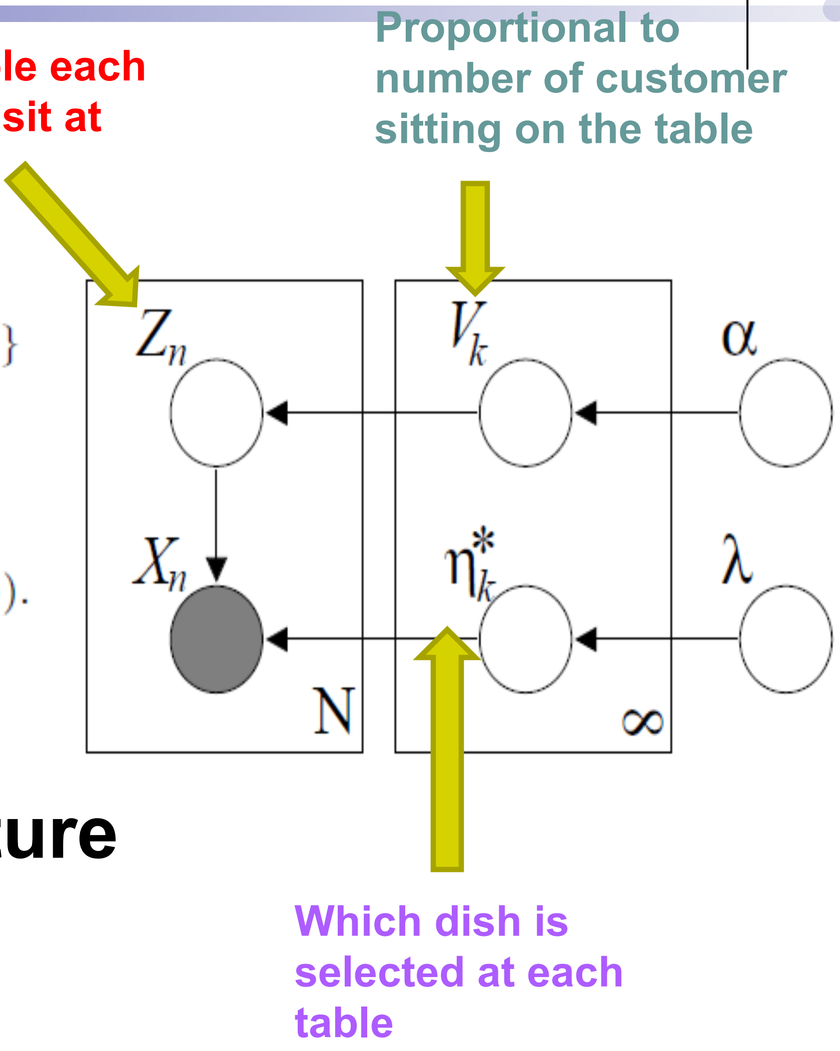
Proportional to selecting a table

Dish on the table



Graphical Model Representation

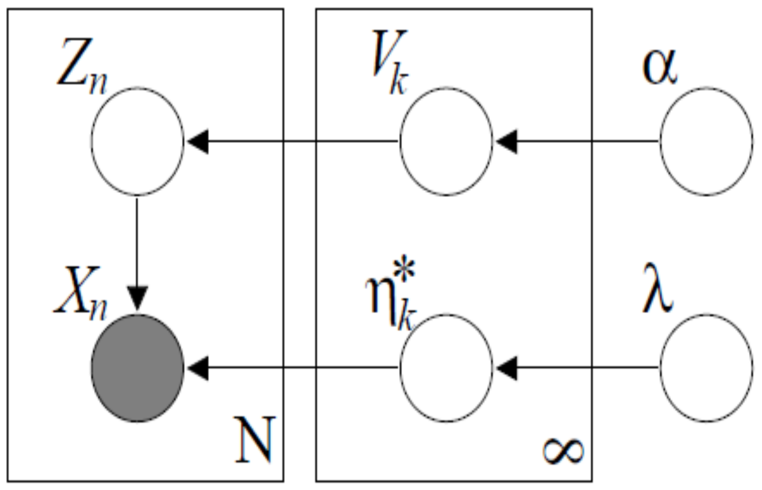
1. Draw $V_i \mid \alpha \sim \mathrm{Beta}(1, \alpha), \quad i = \{1, 2, \ldots\}$
2. Draw $\eta_i^* \mid G_0 \sim G_0, \quad i = \{1, 2, \ldots\}$
3. For the nth data point:
   (a) Draw $Z_n \mid \{v_1, v_2, \ldots\} \sim \mathrm{Mult}(\pi(\mathbf{v}))$.
   (b) Draw $X_n \mid z_n \sim p(x_n \mid \eta_{z_n}^*)$.


Which table each customer sit at
Proportional to number of customer sitting on the table
Z_n
V_k
α
X_n
η_k^*
λ
N
∞
Which dish is selected at each table

Dirichlet Process Mixture Model

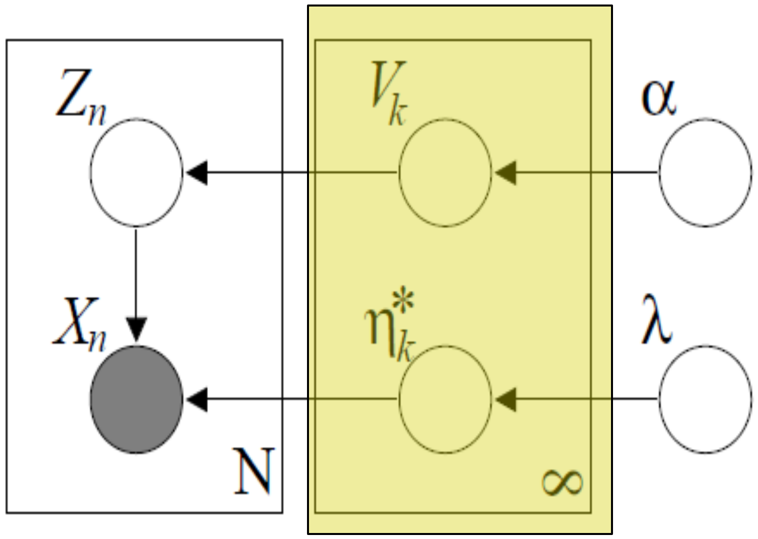
Inference

- Gibbs Sampling:-
  - Sample each of the variable given the rest.
  - Variables to sample are table proportion V_k , table assignment to each customer (Z) and dish at each table η



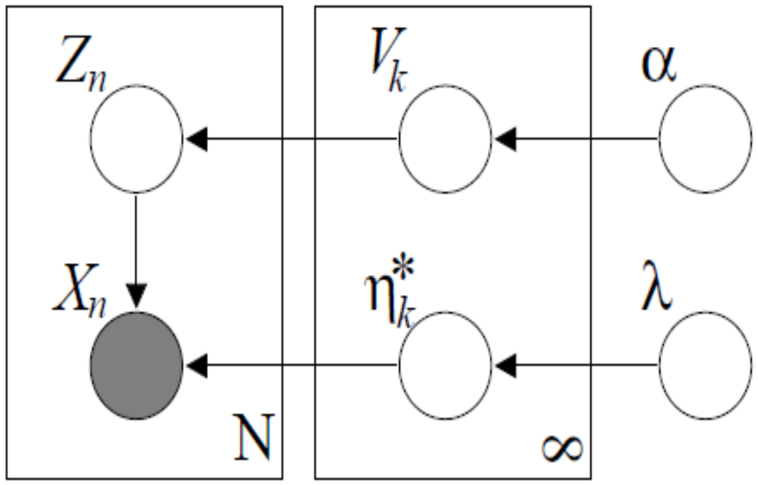
Inference

- Gibbs Sampling:-
  - Sample each of the variable given the rest.
  - Variables to sample are table proportion V_k , table assignment to each customer (Z) and dish at each table η
  - Parallel inference: Easy



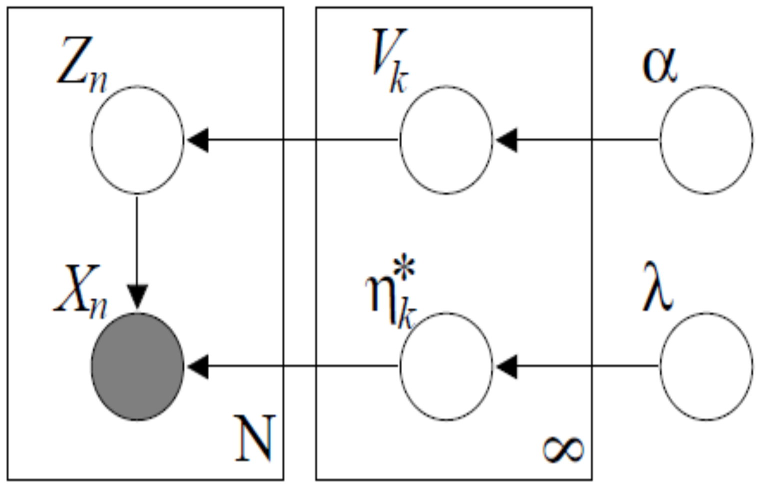
Inference

- Gibbs Sampling:-
  - Sample each of the variable given the rest.
  - Variables to sample are table proportion V_k , table assignment to each customer (Z) and dish at each table η
  - Parallel inference: Easy
  - Poor mixing



Inference

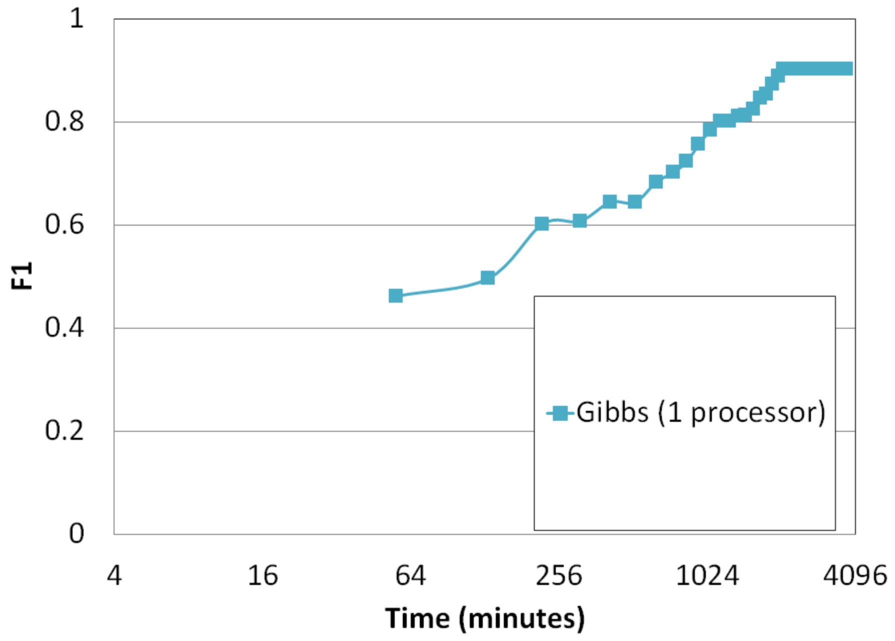
- Collapsed Gibbs Sampler:-
  - Integrate out V_k and η_k
  - Leads to better mixing
  - Parallel inference: Hard



Inference

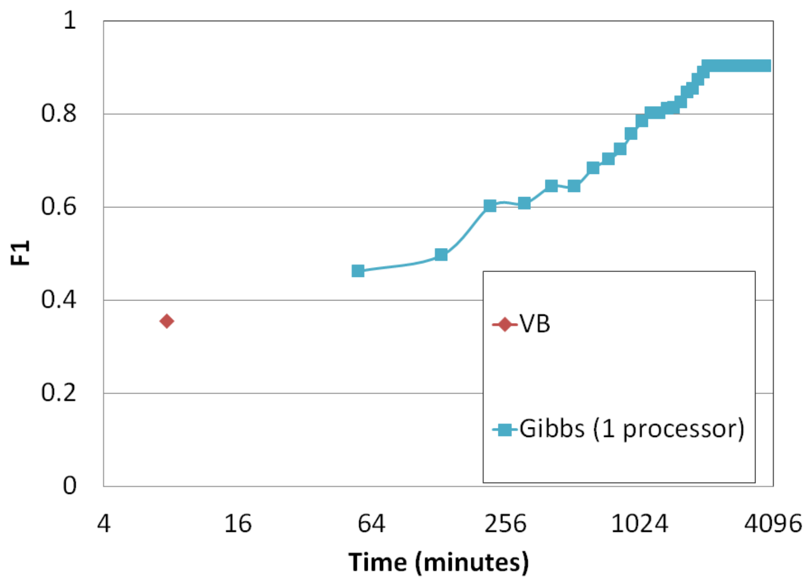
- Collapsed Gibbs suffer from large computational cost

Running Example: 10 million data points to be clustered.



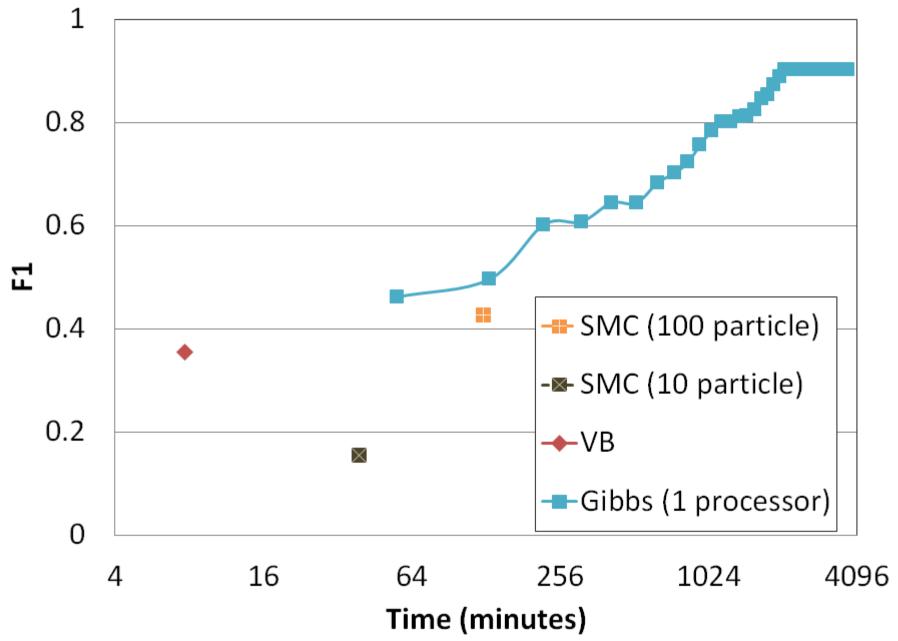
Inference

- Variational Inference
  - Approximate the posterior with a distribution belonging to a more manageable family of distribution
  - Parallel inference: Easy
  - Search within a restricted class of models, looses the expressiveness
  - Typically less accuracy than MCMC methods



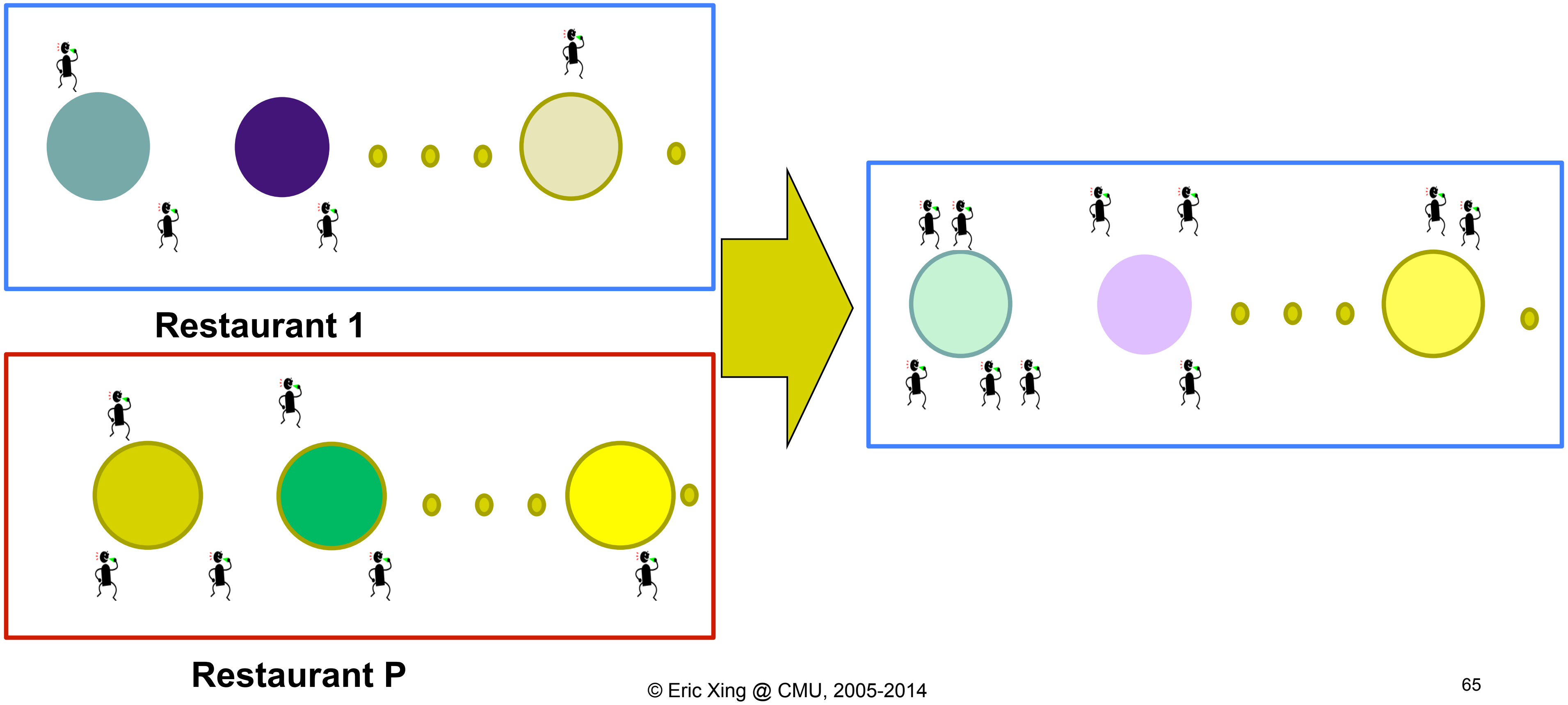
Inference

- Sequential Monte Carlo Method:-
  - Keep a pools of particles, approximate the distribution using weighted combination of the pool
  - Parallel inference: Easy
  - High variance for naïve implementation, needs resampling (MCMC)



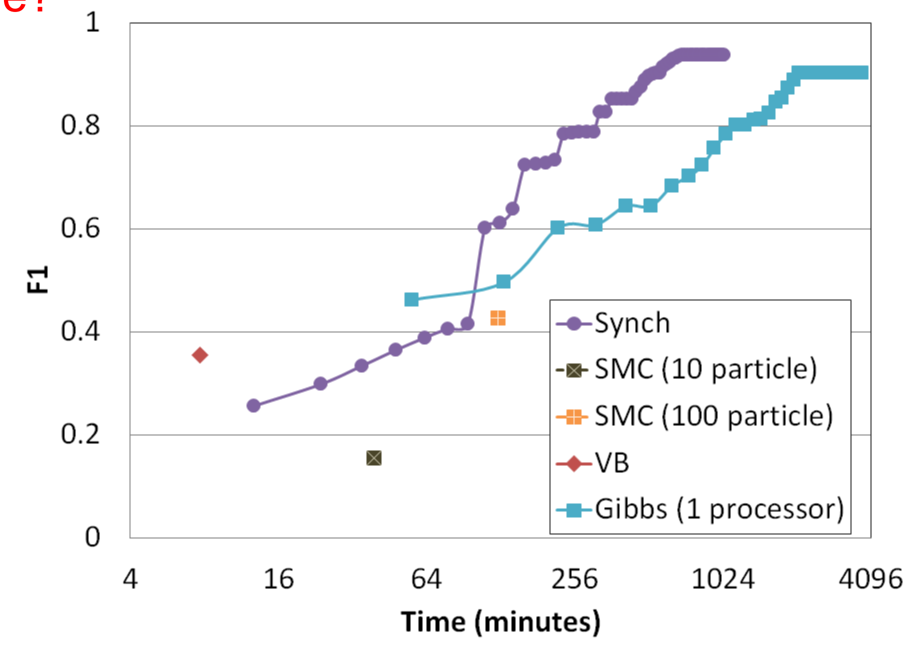
Parallel MCMC

- Naïve
    - Run collapsed sampler on individual core
    - Combine the result approximately !!



Parallel MCMC

- Naïve
  - Run collapsed sampler on individual core
  - Combine the result approximately !!
    - How
    - Why should two newly discovered clustered in two different processor be the same?



Parallel MCMC

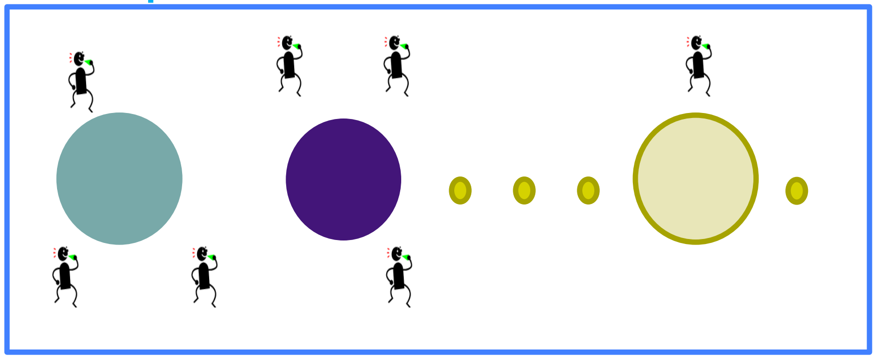
- Idea: Dirichlet Mixture of Dirichlet processes are Dirichlet processes

- Skeptic

(proof coming)

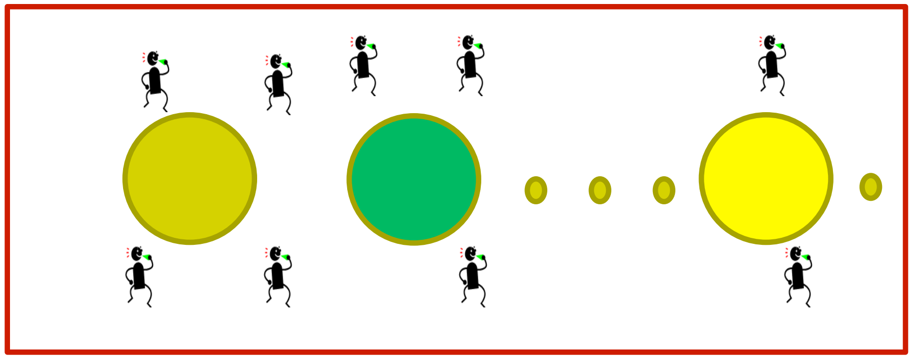
Parallel MCMC

- Idea: Dirichlet Mixture of Dirichlet processes are Dirichlet processes



Restaurant 1

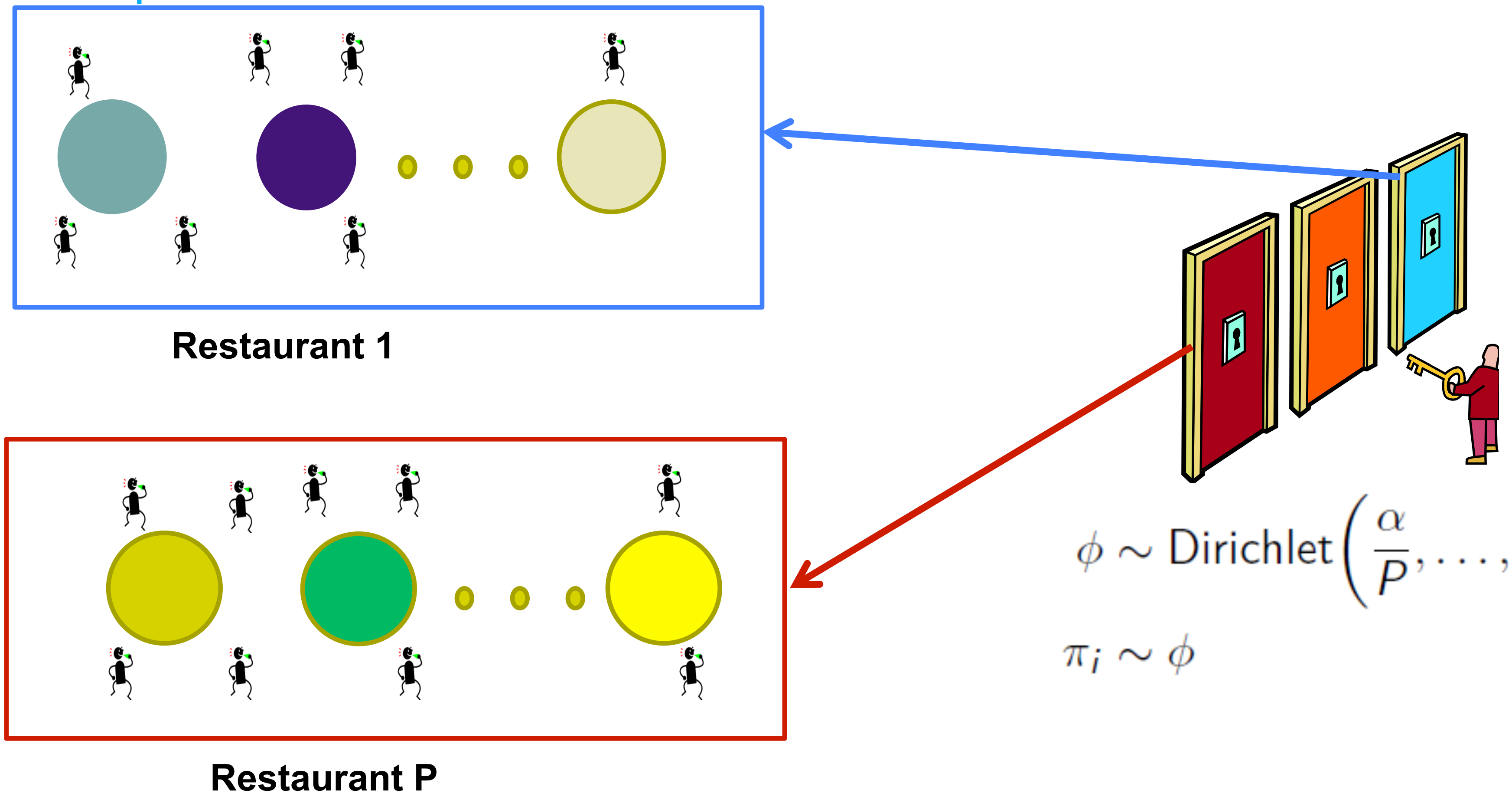
$$D_j \sim \mathrm{DP}\left(\frac{\alpha}{P}, H\right), \quad j = 1, \ldots, P$$



Restaurant P

Parallel MCMC

- Idea: Dirichlet Mixture of Dirichlet processes are Dirichlet processes

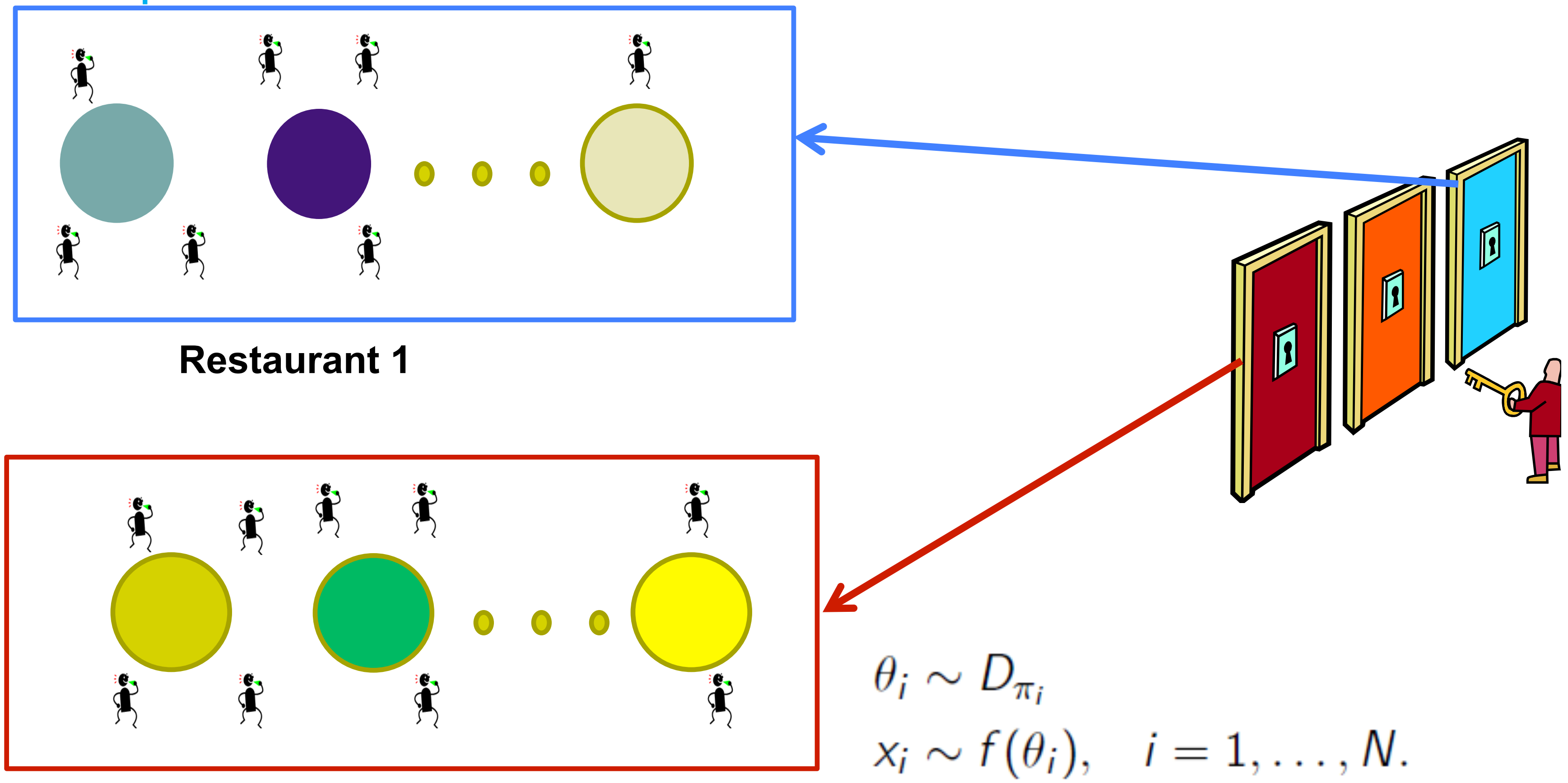


$$\phi \sim \text{Dirichlet}\left(\frac{\alpha}{P}, \ldots, \frac{\alpha}{P}\right)$$

$$\pi_i \sim \phi$$

Parallel MCMC

- Idea: Dirichlet Mixture of Dirichlet processes are Dirichlet processes



$$\theta_i \sim D_{\pi_i}$$
$$x_i \sim f(\theta_i), \quad i = 1, \dots, N.$$

Auxiliary Variable Model For DP

- The generative process is as follows :-

$$\begin{aligned} D_j &\sim \mathrm{DP}\left(\frac{\alpha}{P}, H\right), \quad j = 1, \ldots, P \\ \phi &\sim \mathrm{Dirichlet}\left(\frac{\alpha}{P}, \ldots, \frac{\alpha}{P}\right) \\ \pi_i &\sim \phi \\ \theta_i &\sim D_{\pi_i} \\ x_i &\sim f(\theta_i), \quad i = 1, \ldots, N. \end{aligned}$$

$$\Longleftrightarrow$$

$$\begin{aligned} D &\sim \mathrm{DP}(\alpha, H)\,, \\ \theta_i &\sim D\,, \\ x_i &\sim f(\theta_i) \end{aligned}$$

Proof

- If $G \sim DP(\alpha, G_0)$ and $\theta_1 \sim G$ Then posterior distribution is given by:

$$\theta_{n+1}|\theta_1,\ldots,\theta_n \sim \sum_{l=1}^{n} \frac{1}{n+\alpha}\delta_{\theta_l} + \frac{\alpha}{n+\alpha}G_0$$

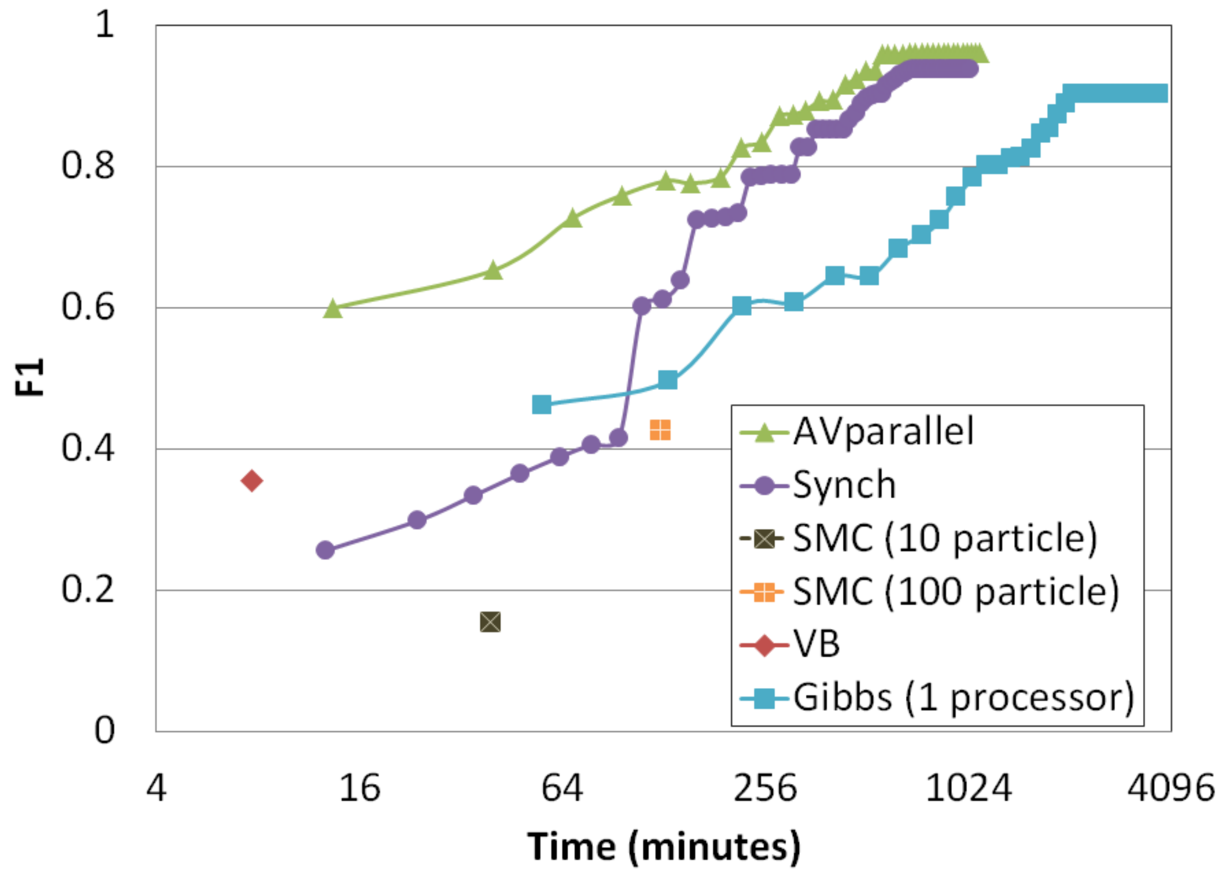
- If $D_j \sim DP(\alpha/P, G_0),\ \phi \sim Dir(\frac{\alpha}{P},\ldots,\frac{\alpha}{P}),\ \pi_i \sim \phi$ and $\theta_i \sim D_{\phi_i}$, Then

$$\begin{aligned}\theta_{n+1}|\theta_1,\ldots,\theta_n \sim &\sum_{j=1}^{P} P(\pi_{n+1}=j|\pi_1,\ldots,\pi_n)\\ &P(\theta_{n+1}|\pi_{n+1}=j,\pi_1,\ldots,\pi_n,\theta_1,\ldots\theta_n,G_0)\\ =&\sum_j \frac{n_j+\alpha/P}{n-1+\alpha}\\ &\left\{\sum_{l=1}^{n}\frac{1}{n_j+\alpha/P}\delta_{\theta_l}\delta_{\pi_l=j}+\frac{\alpha/P}{n_j+\alpha/P}G_0\right\}\\ =&\sum_{l=1}^{n}\frac{1}{n+\alpha}\delta_{\theta_l}+\frac{\alpha}{n+\alpha}G_0\end{aligned}$$

Inference

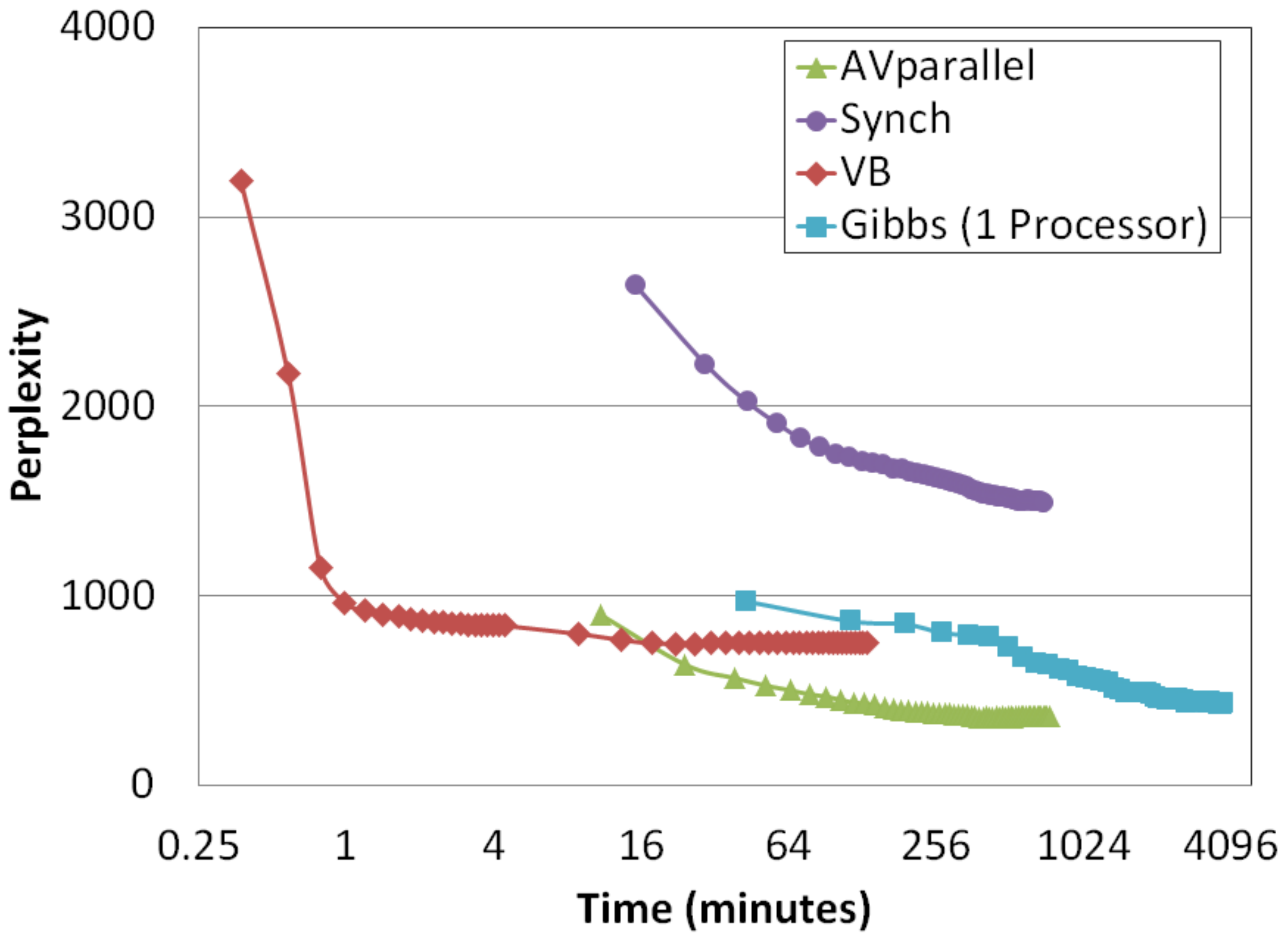
- Conditioned on the Restaurant allocation data are distributed according to P independent Dirichlet process
- Perform local collapsed gibbs sampling on the independent DPs
- For the global parameters perform MH
  - Select a cluster ‘c’ and a processor ‘p’
  - Propose: move ‘c’ to ‘p’
  - Acceptance ratio depends on cluster size
- Can pass the indices of the cluster item.
- Can be done asynchronously without affecting the performance.

Result



1
0.8
0.6
0.4
0.2
0
F1
4
16
64
256
1024
4096
Time (minutes)
AVparallel
Synch
SMC (10 particle)
SMC (100 particle)
VB
Gibbs (1 processor)

Extension to HDP



Take home message

- Naïve parallel inference scheme does not always work
- Utilize structure of the problem: Conditional independence
- Exact parallel inference or bound on error