Entangled Monte Carlo

Seong-Hwan Jun Liangliang Wang Alexandre Bouchard-Côté

University of British Columbia, Vancouver, Canada

Introduction

The sampling stage of Sequential Monte Carlo algorithm is easily parallelizable over multiple processors. Sequential Monte Carlo algorithm depends on the resampling step to *filter* out the samples that are less plausible. The experiments show that omitting the resampling step may lead to degenerate samples. The resampling step is a critical component of Sequential Monte Carlo simulation.
Unfortunately, parallelizing Sequential Monte Carlo over multiple nodes is hard because of the resampling step. The resampling step can cause the particles generated by different computing nodes to be shuffled across different nodes. This requires the transmission of particles between the nodes.
We are interested in parallelization of Sequential Monte Carlo algorithm where the size of the particles grows with the size of the problem.

Background: Sequential Monte Carlo

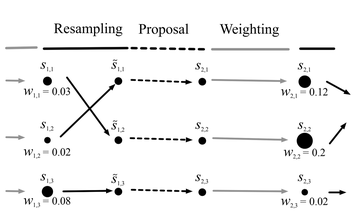
The components of Sequential Monte Carlo algorithm with K particles for $r = 1, \ldots, R$ generations can be broadly categorized as: sample generation, weight computation, and resampling.

At time $r = 0$:

1. Sample $s_{0,j} \sim q_0(s_0)$
2. Compute the weights $w_{0,j} = w(s_{0,j})$ and normalize $\tilde{w}_{0,j} = \tilde{w}(s_{0,j}) \propto w_{0,j}$
3. Resample $\{\tilde{w}_{0,j}, s_{0,j}\}$ to obtain K particles

At time $r \geq 1$:

1. Sample $s_{r,j} \sim q_r(s_r | s_{0:r-1,j})$
2. Compute the weight $w_{r,j}$ and normalize $\tilde{w}_{r,j} \propto w_{r,j}$
3. Resample K times with replacement from $\{\tilde{w}_{r,j}, s_{r,j}\}_{j=1}^{K}$ particles



Sequential Monte Carlo on one node

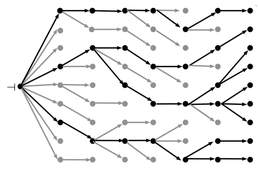
The sample generation step to generate K particles is the most time consuming step in the above algorithm. However, if the simulation is carried out over a single node, the sample generation step can be parallelized over multiple CPUs or GPUs. In this sense, it is already "embarrassingly" parallelizable.

Sequential Monte Carlo on M nodes

At time 0, each node m would be allocated K_m particles such that $\sum_{m=1}^{M} K_m = K$. The number of particles allocated to node m would depend on its capacity. The particle weights needs to be exchanged between the nodes in order to normalize the weights, which would then be used for the resampling step. After the resampling step is completed, it is possible that the number of particles in an arbitrary node m to exceed its capacity, K_m. The surplus particles need to be allocated to other node that are in deficit; this step requires transmission of the particles from the node in surplus to the node in deficit. Therefore, parallelizing Sequential Monte Carlo algorithm over multiple nodes is not so straight forward.

Genealogy

The resampling step uses the normalized particle weights to sample the particles. Similar to bootstrapping, the resampling step is *with replacement*, which means that a particle with high weight may be resampled multiple times. The particles that are resampled survive to the next generation. In essence, the resampling step induces genealogy of particles.



Entangled Monte Carlo

Scalable parallelization of Sequential Monte Carlo over multiple nodes

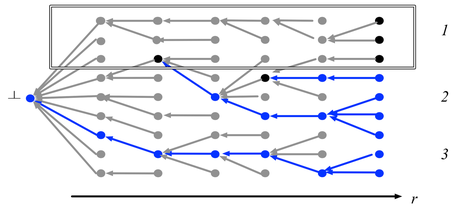
The Entangled Monte Carlo simulation achieves scalable parallelization of Sequential Monte Carlo algorithm by relying on the reconstruction of the particles in lieu of particle transmission. Because we have the particle genealogy where it completely specifies the use of randomness in generating the samples, we can reconstruct sample of any particle by tracing back its genealogy.

Algorithm 1 : EMC$(\alpha, \nu, h, \mathcal{I}_0)$

1: $(\mathscr{F}, \mathscr{G}, \mathscr{H}) \leftarrow$ **entangle**(ν)
2: $s \leftarrow$ **empty-hashtable**
3: $\rho \leftarrow$ **empty-genealogy**
4: **init**(s, w)
5: **for** $r \in \{1, \ldots, R\}$ **do**
6: **exchange**(w_{r-1})
7: **resample**$(w_{r-1}, \rho, \mathcal{I}_{r-1}, \mathscr{G})$
8: $\mathcal{I}_r \leftarrow$ **allocate**$(\rho, \mathcal{I}_{r-1}, \mathscr{H})$
9: **for** $i \in \mathcal{I}_r$ **do**
10: $s(i) \leftarrow$ **reconstruct**$(s, \rho, i, \mathscr{F})$
11: $w_{r,k(i)} \leftarrow \alpha(s(\rho(i)), s(i))$
12: **end for**
13: **end for**
14: **process**(s, w, h)

Local view

For reconstruction of particles to work, each node needs to store the genealogy of all particles. To minimize the storage, each node stores light-weight version of the particles referred to as *compact particles*.



Compact particles

The compact particles have a small memory footprint since they only require to store:

- A *Stochastic map* for each generation
- A parent pointer for each compact particle

A compact particle has miniscule impact on the memory. However, it can be a problem to scaling the Entangled Monte Carlo simulation if the number of compact particles is extremely large. We address this concern by referring to Kingman's coalescent theory.

Stochastic maps

The term stochastic maps comes from the perfect simulation literature based on the seminal work of Propp and Wilson in the late 90's [3]. Given the state space $\mathcal{S}$, a stochastic map F is defined as

$$F : \mathcal{S} \times [0, 1] \to \mathcal{S}$$

Concretely, given an update function $t(U, s)$, where $U \sim \text{Unif}(0, 1)$, we can write $F(s) = t(U, s)$. In the settings of Markov chain, we can start the chain at an arbitrary state $x_0 \in \mathcal{S}$ and obtain the sample after N iterations by sampling $F_1, \ldots, F_N$ maps iid and compose the maps $F_n(\ldots(F_1(x_0))\ldots)$.

Reconstruction

To reconstruct a particle, we need to compose the stochastic maps of the ancestors of the target particle. It can be implemented using a while-loop that traces back until common ancestor is found. The loop is guaranteed to terminate because there is a common source of randomness in which all randomness is generated from.

Algorithm 2 : reconstruct$(s, \rho, i, \mathscr{F} = \{F_i : i \in \mathcal{I}\})$

1: $F \leftarrow I$
2: **while** $(s(i) = \text{nil})$ **do**
3: $F \leftarrow F \circ F_i$
4: $i \leftarrow \rho(i)$
5: **end while**
6: **return** $F(s(i))$

Particle allocation

The particle allocation step achieves load balancing by ensuring that each node generates samples according to its capacity K_m. We recommend greedy allocation schemes where each node tries to keep as many particles as possible before allocating the surplus to other nodes. For the experiments, we have tried the following schemes:

- FIRST-OPEN: assignment based on a pre-complied list of preferred nodes.
- MOST-AVAILABLE: based on the capacity remaining.
- RANDOM: randomly assigns to a node with deficit. This method spreads the particles evenly across the nodes, which can shorten the reconstruction time.

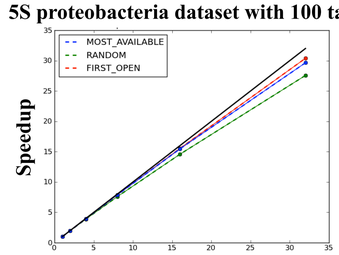
Kingman's coalescent

The potential problems with the Entangled Monte Carlo simulation is that of the memory usage and the expected reconstruction time. The reconstruction time depends on the allocation scheme as well as the transition kernel, which is the subject of future work. Here, we refer to Kingman's coalescent theory, which provides $(1 - 1/k)/(1 - 1/K)$ as the expected time spent waiting for the last k copies to coalesce ([2, 1]).
If there are $K = 1,00,000$ particles in the simulation and if $k = 2$,

$$\frac{(1 - 1/2)}{1 - 1/1,000,000} = 0.5$$

so we expect to wait 50%, of the time for the last two particles to coalesce. In other words, $999,998$ particles would have coalesced in the first 50% of the time.

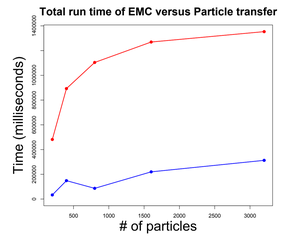
5S proteobacteria dataset with 100 taxa



Denoting N_1 as the total number of times the stochastic maps are applied and N_M as the total number of times the stochastic maps are applied in an Entangled Monte Carlo simulation involving M nodes, the speedup factor is computed as, $S_M = M\frac{N_1}{N_M}$.
This experiment verifies that the reconstruction rarely traces deep and the allocation schemes suggested perform comparably to one another.

Experiments

The total runtime of Entangled Monte Carlo is compared against total run time of Sequential Monte Carlo as the number of particles is increased. The Entangled Monte Carlo (blue line) clearly outperforms Sequential Monte Carlo (red line).



This is attributable to the fact that:

1. CPU cycles faster than communication via network protocol and,
2. Reconstruction rarely traces deep as predicted by Kingman's coalescent.

Conclusion

The Entangled Monte Carlo achieves parallelization of Sequential Monte Carlo independent of the particle size by reconstructing the particles from the particle genealogy. The future work involves extending the method to implement storage of particle genealogy using *distributed hash table*, which will allow Entangled Monte Carlo to be extended to be applied to situation such as BOINC.

References

[1] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2003.
[2] J. F. C. Kingman. On the Genealogy of Large Populations. *Journal of Applied Probability*, 19:27–43, 1982.
[3] J. Propp and D. Wilson. Coupling from the past: a user's guide, 1997.