Divide-and-Conquer Monte Carlo Fusion
A method of unifying distributed analyses

Ryan Chan [1,3], Murray Pollock [1,2], Adam Johansen [1,3], Gareth Roberts [1,3]

[1] University of Warwick, [2] Newcastle University, [3] The Alan Turing Institute

Generalised Monte Carlo Fusion

Provides theory to carry out *exact* inference for the target:

$$\pi(\boldsymbol{x}) \propto f_1(\boldsymbol{x}) \cdots f_C(\boldsymbol{x}) = \prod_{c=1}^{C} f_c(\boldsymbol{x}). \tag{1}$$

Uses *importance sampling* on the extended target distribution:

$$g(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}) \propto \prod_{c=1}^{C} \left[f_c^2(\boldsymbol{x}^{(c)}) \cdot p_c(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}) \cdot \frac{1}{f_c(\boldsymbol{y})} \right]. \tag{2}$$

Let $p_c(\boldsymbol{y} \mid \boldsymbol{x}^{(c)})$ be a transition density of a stochastic process with stationary distribution $f_c^2(\boldsymbol{x})$, then (2) admits π as a marginal for $\boldsymbol{y}$.
Proposal distribution:

$$h(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}) \propto \prod_{c=1}^{C} \left[f_c\left(\boldsymbol{x}^{(c)}\right) \right] \cdot \exp\left\{ -\frac{(\boldsymbol{y} - \tilde{\boldsymbol{x}})^{\intercal} \Lambda^{-1} (\boldsymbol{y} - \tilde{\boldsymbol{x}})}{2T} \right\}, \tag{3}$$

where

$$\tilde{\boldsymbol{x}} := \left(\sum_{c=1}^{C} \boldsymbol{\Lambda}_c^{-1} \right)^{-1} \left(\sum_{c=1}^{C} \boldsymbol{\Lambda}_c^{-1} \boldsymbol{x}^{(c)} \right), \qquad \boldsymbol{\Lambda}^{-1} := \sum_{c=1}^{C} \boldsymbol{\Lambda}_c^{-1}. \tag{4}$$

Considering the transition probability of a d-dimensional double Langevin diffusion process, then under certain mild conditions,

$$\frac{g\left(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}\right)}{h\left(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}\right)} \propto \rho_0 \cdot \rho_1 =: w'\left(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}\right), \tag{5}$$

where

$$\rho_0(\boldsymbol{x}^{(1:C)}) := \exp\left\{ -\sum_{c=1}^{C} \frac{(\tilde{\boldsymbol{x}} - \boldsymbol{x}^{(c)})^{\intercal} \boldsymbol{\Lambda}_c^{-1} (\tilde{\boldsymbol{x}} - \boldsymbol{x}^{(c)})}{2T} \right\}, \tag{6}$$

$$\rho_1(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}) := \prod_{c=1}^{C} \mathbb{E}_{\mathbb{W}_{\Lambda_c}} \left[\exp\left\{ -\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t \right\} \right], \tag{7}$$

and

$$\phi_c(\boldsymbol{x}) := \frac{1}{2}\left(\nabla \log f_c(\boldsymbol{x})^{\intercal} \boldsymbol{\Lambda}_c \nabla \log f_c(\boldsymbol{x}) + \mathrm{Tr}\left(\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x}) \right) \right), \tag{8}$$

where $\mathbb{W}_{\boldsymbol{\Lambda}_c}$ denotes the law of a Brownian bridge $\{\boldsymbol{X}_t^{(c)}, t \in [0, T]\}$ with $\boldsymbol{X}_0^{(c)} := \boldsymbol{x}^{(c)}$, $\boldsymbol{X}_T^{(c)} := \boldsymbol{y}$ and covariance matrix $\boldsymbol{\Lambda}_c$.

Algorithm 1 Generalised Monte Carlo Fusion

1. Simulate a proposal $\boldsymbol{y}$ from h:
   a) For $c = 1, \ldots, C$, simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$ and calculate $\tilde{\boldsymbol{x}}$
   b) Simulate $\boldsymbol{y} \sim \mathcal{N}_d(\tilde{\boldsymbol{x}}, T\Lambda)$
2. Assign unnormalised weight $w'\left(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}\right)$

Robustness to correlation

Let $\pi \propto f_1 f_2$, where $f_c \sim \mathcal{N}_2(\boldsymbol{0}, \Sigma)$ and $\Sigma = \begin{pmatrix} 1.0 & \rho_{\text{corr}} \\ \rho_{\text{corr}} & 1.0 \end{pmatrix}$.

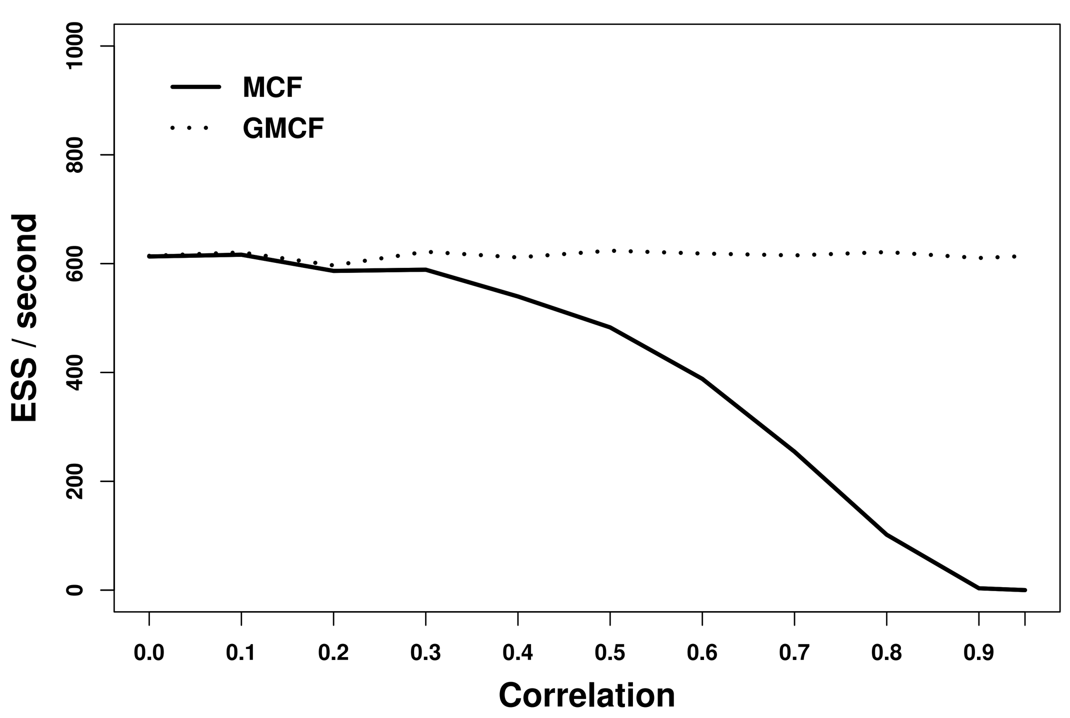

Figure 1: ESS per second (averaged over 50 runs) when contrasting Monte Carlo Fusion [Dai et al., 2019] (i.e. $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c = 1, \ldots, C$) and Generalised Monte Carlo Fusion, along with increasing sub-posterior correlation, ρ_{corr}.

Divide-and-Conquer Monte Carlo Fusion

Problem: Fusion becomes inefficient as the number of sub-posteriors increases.

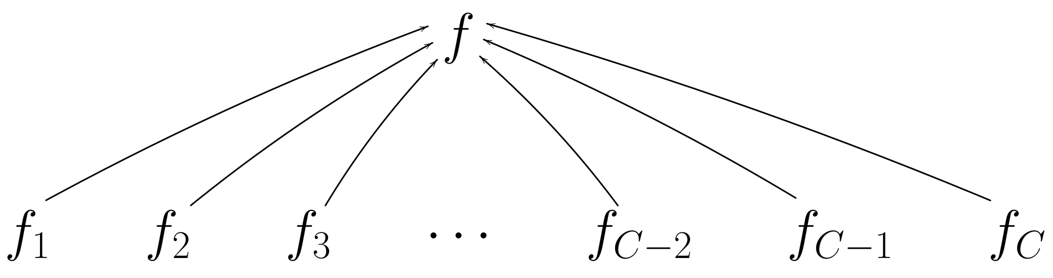

Figure 2: The 'fork-and-join' approach.

However, we can adopt a *divide-and-conquer* approach:

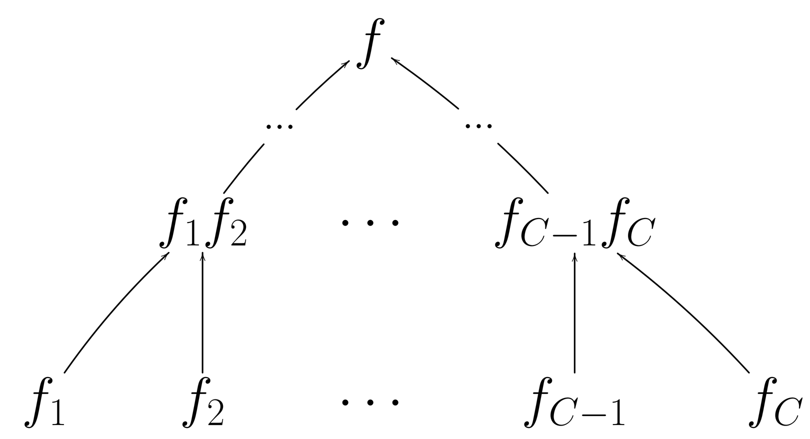

Figure 3: A balanced-binary tree.

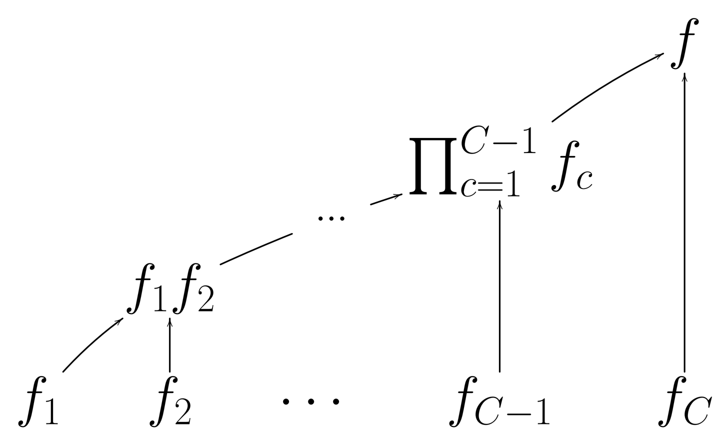

Figure 4: A progressive tree.

Robustness to number of sub-posteriors

$\pi \propto \prod_{c=1}^{C} f_c$, where $f_c \sim \mathcal{N}(0, C)$ for $c = 1, \ldots, C$. To compare methods, use integrated absolute distance (where $\hat{f}(\boldsymbol{x}_j)$ is target density):

$$IAD = \frac{1}{2d} \sum_{j=1}^{d} \int \left| \hat{f}(\boldsymbol{x}_j) - f(\boldsymbol{x}_j) \right| \mathrm{d}x_j \in [0, 1].$$

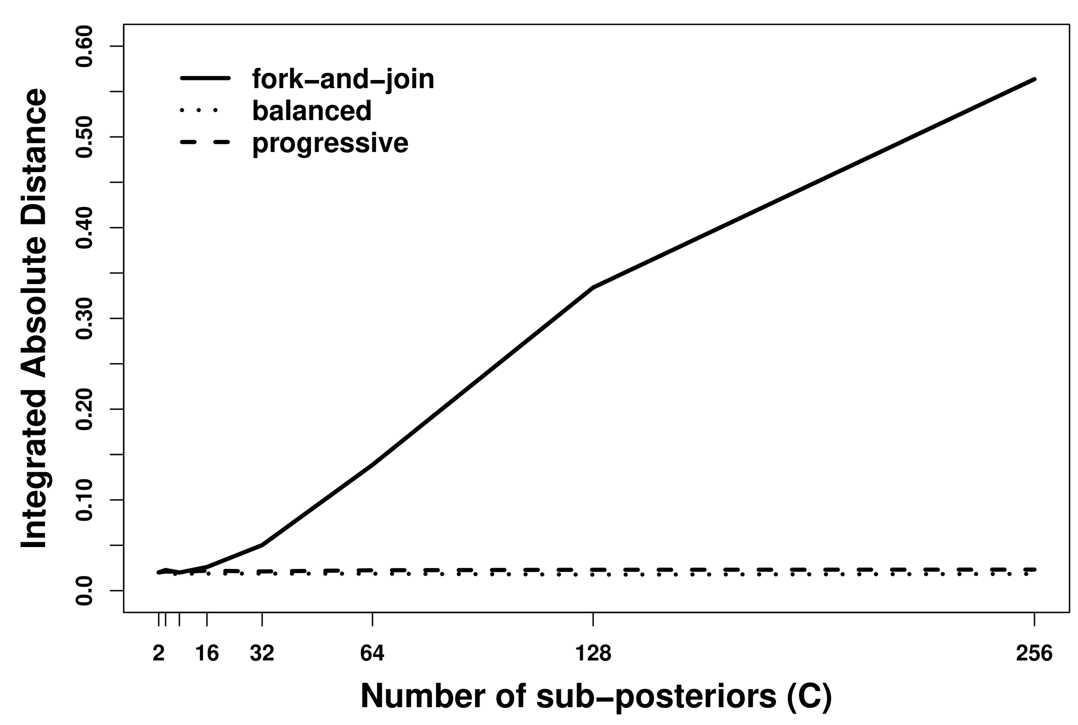

Figure 5: IAD (averaged over 50 runs) when contrasting different tree hierarchies.

Logistic regression example

Logistic regression model with credit card data to predict default on loans ($n = 50,000$, $d = 5$).

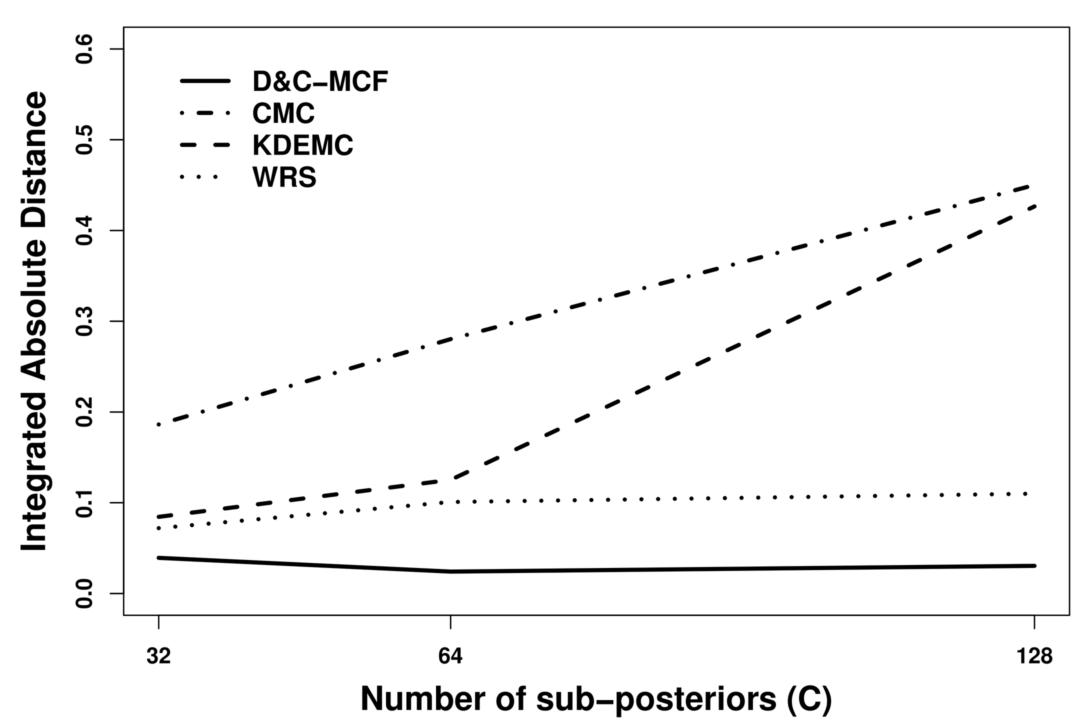

Figure 6: IAD for different number of sub-posteriors for different methods for unifying distributed analyses.

References

- Chan, R.S.Y., Johansen, A.M., Pollock, M., and Roberts, G.O. 2021. *Divide-and-Conquer Monte Carlo Fusion.* Submitted. arXiv:2110.07265
- Dai, Hongsheng, Murray Pollock, and Gareth Roberts. *Monte Carlo Fusion.* Journal of Applied Probability (56)1. 2019: pp. 174-191.