A Distributed Plug-and-Play MCMC Sampler for Large Imaging Inverse Problems

Maxime Bouton[1] Pierre-Antoine Thouvenin[1] Audrey Repetti[2, 3] Pierre Chainais[1]

[1]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
[3]Maxwell Institute for Mathematical Sciences, Edinburgh EH8 9BT, UK

Colloque GRETSI - 28th August 2025

High-Dimensional Inverse Problems

- Inputs:
  - Observations $\boldsymbol{y} \in \mathbb{R}^M$ (e.g., noisy, degraded)
  - Latent image $\overline{\boldsymbol{x}} \in \mathbb{R}^N$
  - $M, N \sim 10^7$

$\boldsymbol{y}$

$\overline{\boldsymbol{x}}$

- Measurement model: $\boldsymbol{y} = \mathcal{A}(\boldsymbol{H}\overline{\boldsymbol{x}})$
- Posterior: $\pi(\boldsymbol{x}|\boldsymbol{y}) \propto \exp(-\phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x}))p(\boldsymbol{x})$

Motivations

1. **Uncertainty quantification** $\rightsquigarrow$ Bayesian Inference (e.g. MCMC)

2. **Reconstruction quality**
   - Limitation of traditional hand-crafted priors (e.g., TV)
   - Learned deep denoisers $\rightsquigarrow$ PnP methods

3. **Very large problems**
   - Memory load
   - Long runtime

   $\rightsquigarrow$ distributed computing

Plug-and-Play MCMC

Existing PnP-MCMC samplers:

- PnP-ULA [Laumont et al. (2022)]
- PnP-SGS [Coeurdoux et al. (2024)]
- RED [Faye et al. (2024)]

PnP-ULA

Langevin dynamics with deep denoiser as prior

$$
\begin{aligned}
\boldsymbol{z}^{(t+1)} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N) \\
\boldsymbol{x}^{(t+1)} &= \boldsymbol{x}^{(t)} - \delta \boldsymbol{H}^* \nabla \phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x}^{(t)}) + \sqrt{2\delta}\boldsymbol{z}^{(t+1)} \\
&\quad + \frac{\alpha\delta}{\epsilon}\left(D_\epsilon(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\right) + \frac{\delta}{\lambda}\left(\Pi_{\mathcal{C}}(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\right),
\end{aligned}
$$

with D_ϵ pre-trained (deep) Gaussian denoiser, variance ϵ
$\Pi_{\mathcal{C}}$ projection onto $\mathcal{C} \subset \mathbb{R}^N$, non-empty compact convex set

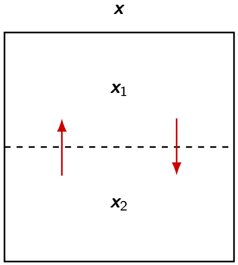
Challenges in High-Dimensional MCMC

- Sequential nature of MCMC limits scalability
  $\rightsquigarrow$ can be challenging to run a single chain
- Distributed computing offers a solution

Single Program Multiple Data (SPMD) Paradigm

- Same program running on all workers
- Different subsets of data on each worker

Efficiency $\Rightarrow$ limited communications



Towards a distributed sampler: focus on operators

Distributing Pnp-ULA over B workers:

$$\begin{aligned}\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \delta \boldsymbol{H}^* \nabla \phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x}^{(t)}) + \sqrt{2\delta}\boldsymbol{z}^{(t+1)} \\ + \frac{\alpha\delta}{\epsilon}\big(D_\epsilon(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\big) + \frac{\delta}{\lambda}\big(\Pi_{\mathcal{C}}(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\big)\end{aligned}$$

Exploits locality property

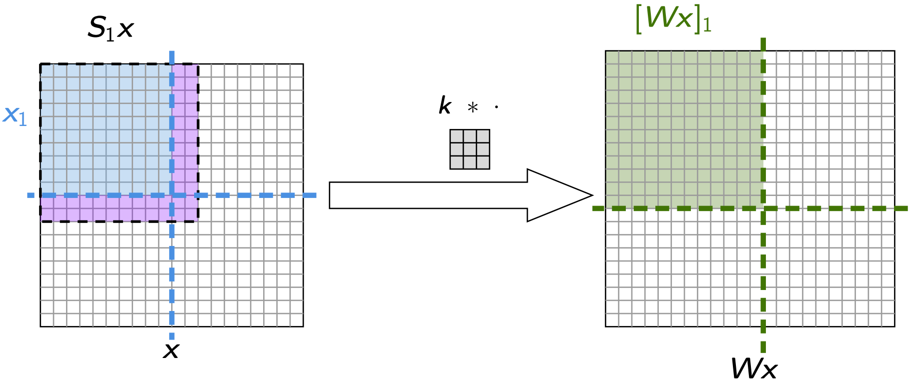
f *local* operator $\Leftrightarrow [f(\boldsymbol{x})]_i = \widetilde{f}_i(\boldsymbol{x}_{[i]})$, $\quad \boldsymbol{x}_{[i]}$: "small" subset of $\boldsymbol{x}$

Assumptions: $\boldsymbol{x} = (\boldsymbol{x}_b)_{1 \leq b \leq B}$

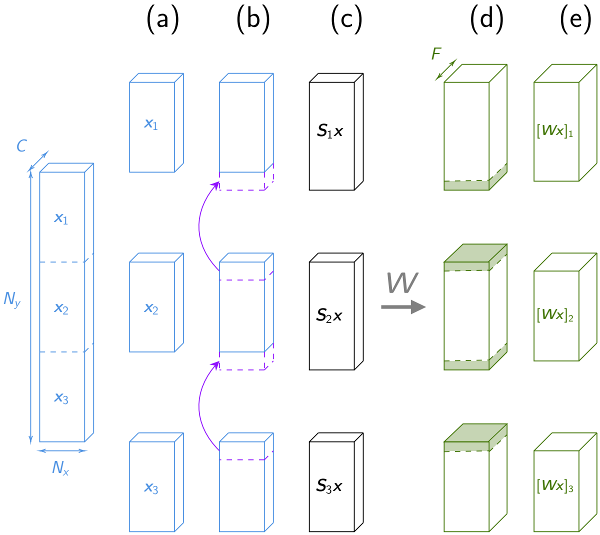
- $\boldsymbol{H}$ local and $\phi_{\boldsymbol{y}}$ block-additively separable
  $\Rightarrow (\boldsymbol{y}_b)_{1 \leq b \leq B}$ s.t. $\boldsymbol{y}_b \perp\!\!\!\perp \boldsymbol{y}_{b'} \mid \boldsymbol{x}$
- $D_\epsilon, \Pi_{\mathcal{C}}$ local $\Rightarrow$ limited communications

CNNs as distributable priors

- Exploit state-of-the-art "on-the-shelf" denoising CNNs (e.g. DDFB [Repetti et al. (2022)])
- Mainly involve convolution layers
  - Convolution is a *local* operation
  - Small kernel size $\rightsquigarrow$ small borders to communicate



Distributed convolution



Input: $\boldsymbol{y} \in \mathbb{R}^M, \epsilon > 0, \alpha > 0, \lambda, \delta$

for *each worker* $b \in \{1, \dots, B\}$ **do** in parallel

  Load and store $\boldsymbol{y}_b \in \mathbb{R}^{M_b}$

  Initialize $\boldsymbol{x}_b^{(0)} \in \mathbb{R}^{N_b}$

  for $t = 0$ **to** $T-1$ **do**

    `// Communicate to compute`

    $$\boldsymbol{g}_b{}^{(t)} = \frac{1}{\sigma^2}\left[\boldsymbol{H}^*\left(\boldsymbol{H}\boldsymbol{x}^{(t)} - \boldsymbol{y}\right)\right]_b \qquad = \left[\boldsymbol{H}^*\nabla\phi_{\boldsymbol{y}}\left(\boldsymbol{H}\boldsymbol{x}^{(t)}\right)\right]_b$$

    `// Communicate within each layer of DDFB to compute`

    $$\boldsymbol{d}_b{}^{(t)} = \left[D_\epsilon(\boldsymbol{x}^{(t)})\right]_b$$

    `// Local update`

    Draw $\boldsymbol{z}_b{}^{(t+1)} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I}_{N_b})$

    $$\boldsymbol{x}_b{}^{(t+1)} = \boldsymbol{x}_b{}^{(t)} - \delta\boldsymbol{g}_b{}^{(t)} + \frac{\alpha\delta}{\epsilon}\left(\boldsymbol{d}_b{}^{(t)} - \boldsymbol{x}_b{}^{(t)}\right) + \frac{\delta}{\lambda}\left(\Pi_{\mathcal{C}_b}(\boldsymbol{x}_b{}^{(t)}) - \boldsymbol{x}_b{}^{(t)}\right) + \sqrt{2\delta}\,\boldsymbol{z}_b{}^{(t+1)}$$

  end

end

Strong scalability results : Gaussian inpainting

$N = (3 \times 1024 \times 1024)$, 80% missing pixels, $\sigma = 0.03$ and DDFB ($K = 19$)

(a) $\boldsymbol{y}$ (b) $\overline{x}$ (c) $\widehat{x}_{\mathrm{MMSE}}$ ($B = 4$) (d) 95% IC ($B = 4$)

#GPUs	Runtime (ms)	Speedup	rSNR	SSIM
1	199 ± 0.32	1	16.29	0.78
2	137 ± 0.45	1.45	16.33	0.78
4	105 ± 0.94	1.90	16.30	0.78

Conclusion

Conclusion

This distributed sampler:

- **Equivalent** to sequential sampler (up to random number generator)
- Can handle **larger problems** + **Speed up**

Perspectives

- Other distributed schemes [Galerne et al. (2024)]
- Different types of deep denoisers and transition kernels
- Structures beyond images (e.g., graphs)
- Explore asynchronous versions

References

Coeurdoux, Florentin et al. (2024). : Plug-and-Play Split Gibbs Sampler: Embedding Deep Generative Priors in Bayesian Inference. *IEEE Transactions on Image Processing* 33, pp. 3496–3507.

Faye, Elhadji C. et al. (Feb. 2024). *Regularization by Denoising: Bayesian Model and Langevin-within-split Gibbs Sampling*.

Galerne, Bruno et al. (2024). : Scaling Painting Style Transfer. *Computer Graphics Forum* 43.4, e15155.

Laumont, Rémi et al. (June 2022). : Bayesian Imaging Using Plug & Play Priors: When Langevin Meets Tweedie. *SIAM Journal on Imaging Sciences* 15.2, pp. 701–737.

Repetti, Audrey et al. (Aug. 29, 2022). : Dual Forward-Backward Unfolded Network for Flexible Plug-and-Play. Belgrade, Serbia: IEEE, pp. 957–961.

Weak scalability results

#GPUs (image size)	Runtime (ms)	Efficiency
1 $\left(3 \times 1024^2\right)$	199 ± 0.32	1
2 $\left(3 \times 1448^2\right)$	239 ± 8.86	0.83
4 $\left(3 \times 2048^2\right)$	284 ± 31.43	0.70

DDFB

Unrolled from a dual forward-backward algorithm [Repetti et al. (2022)].

$$D_\epsilon(\boldsymbol{v}) = \mathrm{proj}_{[0,1]^N}\left(\boldsymbol{v} - \gamma_K \boldsymbol{W}_K^* G_{\epsilon,\boldsymbol{v}}(\boldsymbol{W}_K \boldsymbol{v})\right),$$

with $G_{\epsilon,\boldsymbol{v}} = T_{K-1,\epsilon,\boldsymbol{v}} \circ \cdots \circ T_{1,\epsilon,\boldsymbol{v}}$ such that

$$T_{k,\epsilon,\boldsymbol{v}}(\boldsymbol{u}) = \mathcal{HT}_{\sqrt{\epsilon}}\Big(\boldsymbol{u} + \gamma_k \boldsymbol{W}_k\, \mathrm{proj}_{[0,1]^N}(\boldsymbol{v} - \boldsymbol{W}_k^* \boldsymbol{u})\Big)$$