Speed up population Bayesian inference by combining cross-chain warmup and within-chain parallelization

Yi Zhang[1], William R. Gillespie[1], Ben Bales[2], Aki Vehtari[3] 1. Metrum Research Group, 2. Columbia University, 3. Aalto University

Objectives

Automate Markov Chain Monte Carlo (MCMC) convergence assessment to adaptively transition from warmup to sampling.
Improve adaptive tuning of Hamiltonian Monte Carlo (HMC) parameters.
Speed up population model inference by combining new warmup method with within-chain parallelization[4][6].

Cross-chain warmup

Researchers have long been seeking a measure to evaluate MCMC warmup quality. A common practice of MCMC toolkits such as Stan[1] is to prescribe a fixed number of warmup iterations, of which the efficiency/sufficiency is revealed only *at the end* of simulation, through quantities such as potential scale reduction coefficients ($\hat{R}$) and effective sample sizes (ESS)[5]. In general there is yet an established method for dynamical warmup assessment before transitioning to post-warmup sampling. For that we propose the following algorithm:

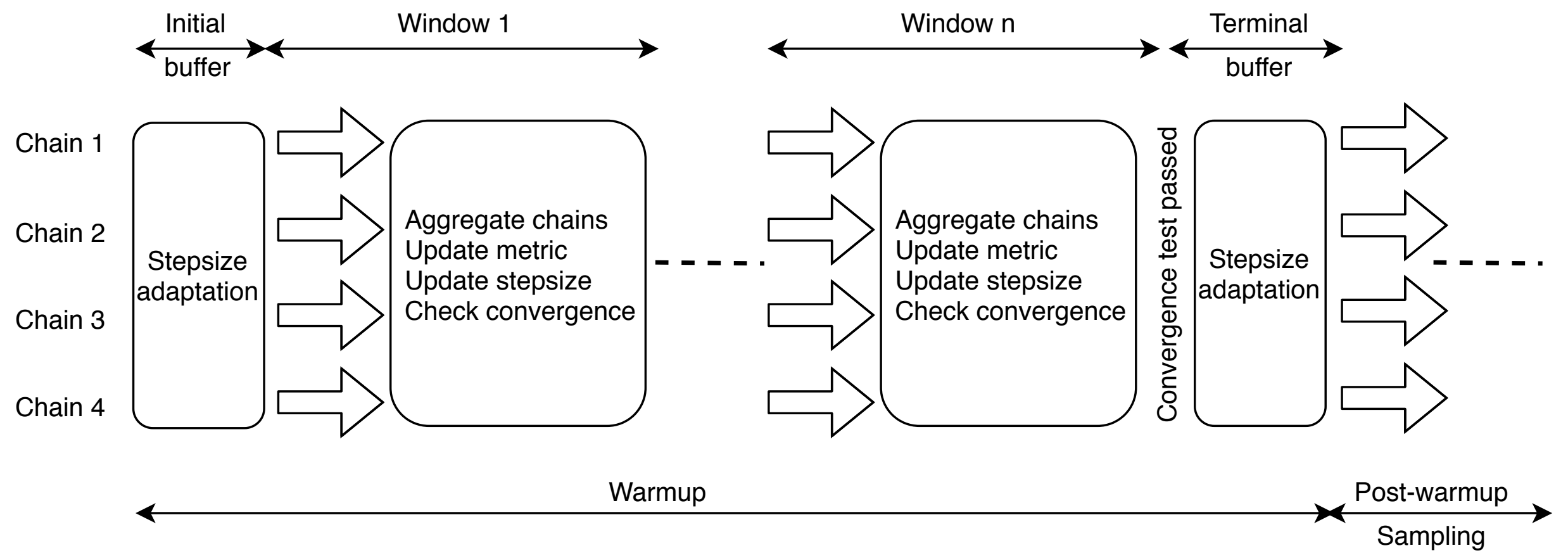

Figure 1: Proposed *cross-chain* warmup algorithm

1. With a fixed window size w, initiate warmup with stepsize adaptation.
2. At the end of a window, aggregate joint posterior probability from all the chains and calculate corresponding $\hat{R}$ and ESS. For example, with default window size $w = 100$, when warmup reaches iteration 200, calculate $\hat{R}^i$ and ESS^i for $i = 1, 2$, so that $\hat{R}^1$ and ESS^1 are based on warmup iteration 1 to 200, and $\hat{R}^2$ and ESS^2 are based on warmup iteration 101 to 200.
3. At the end of window n with predefined target value $\hat{R}^0$ and ESS^0, from $1, \dots, n$, select j with maximum ESS^j and calculate a new metric using samples from corresponding windows. Determine *convergence* by checking if $\hat{R}^j < \hat{R}^0$ and $\text{ESS}^j > \text{ESS}^0$. If converges, move to post-warmup sampling, otherwise repeat step 2.

Benchmarks are performed with different target ESS and regular Stan run (1000 warmup iterations). We run each setup with 10 random seeds and collect average (barplot) and standard deviation (error bar) of the following quantities.

total number of leapfrog integration steps in warmup
total number of leapfrog integration steps in sampling
number of leapfrog integration steps in per each warmup iteration
number of leapfrog integration steps in per each sampling iteration
minimum ESS_{bulk} per iteration
minimum ESS_{tail} per iteration
minimum ESS_{bulk} per leapfrog step
minimum ESS_{tail} per leapfrog step
maximum wall time (in seconds)

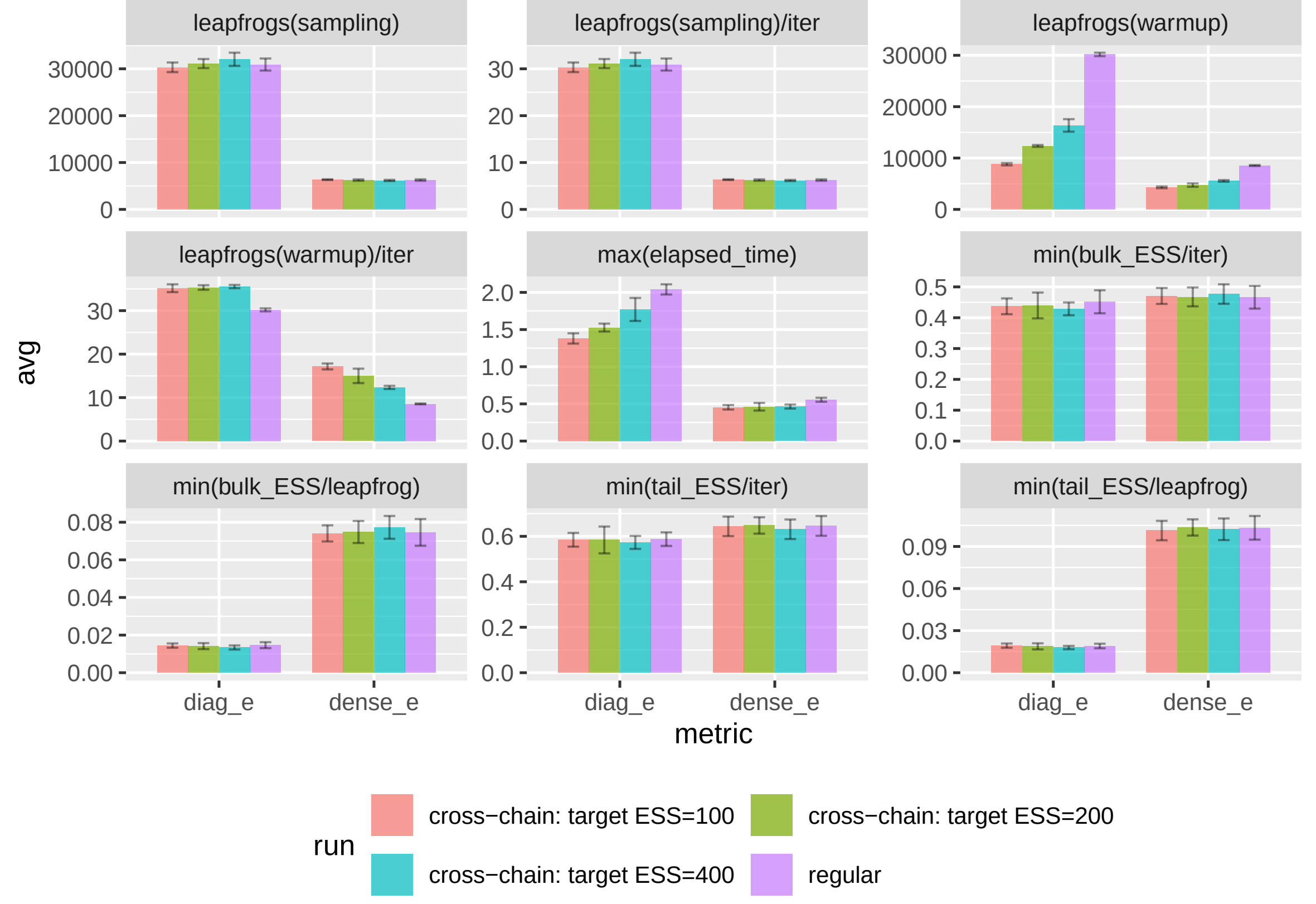

Figure 2: Cross-chain warmup performance: arK model[3]

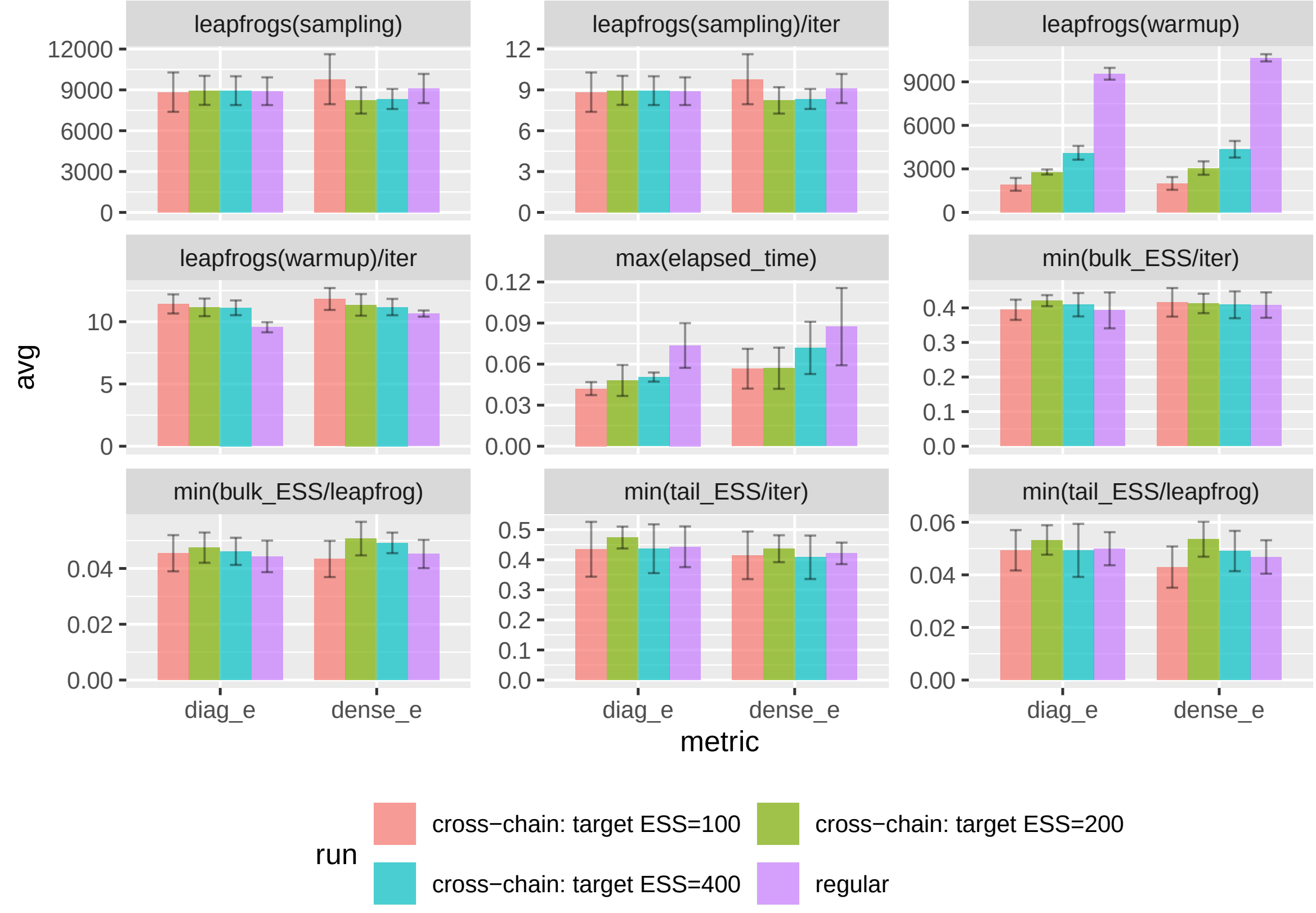

Figure 3: Cross-chain warmup performance: eight schools model[3]

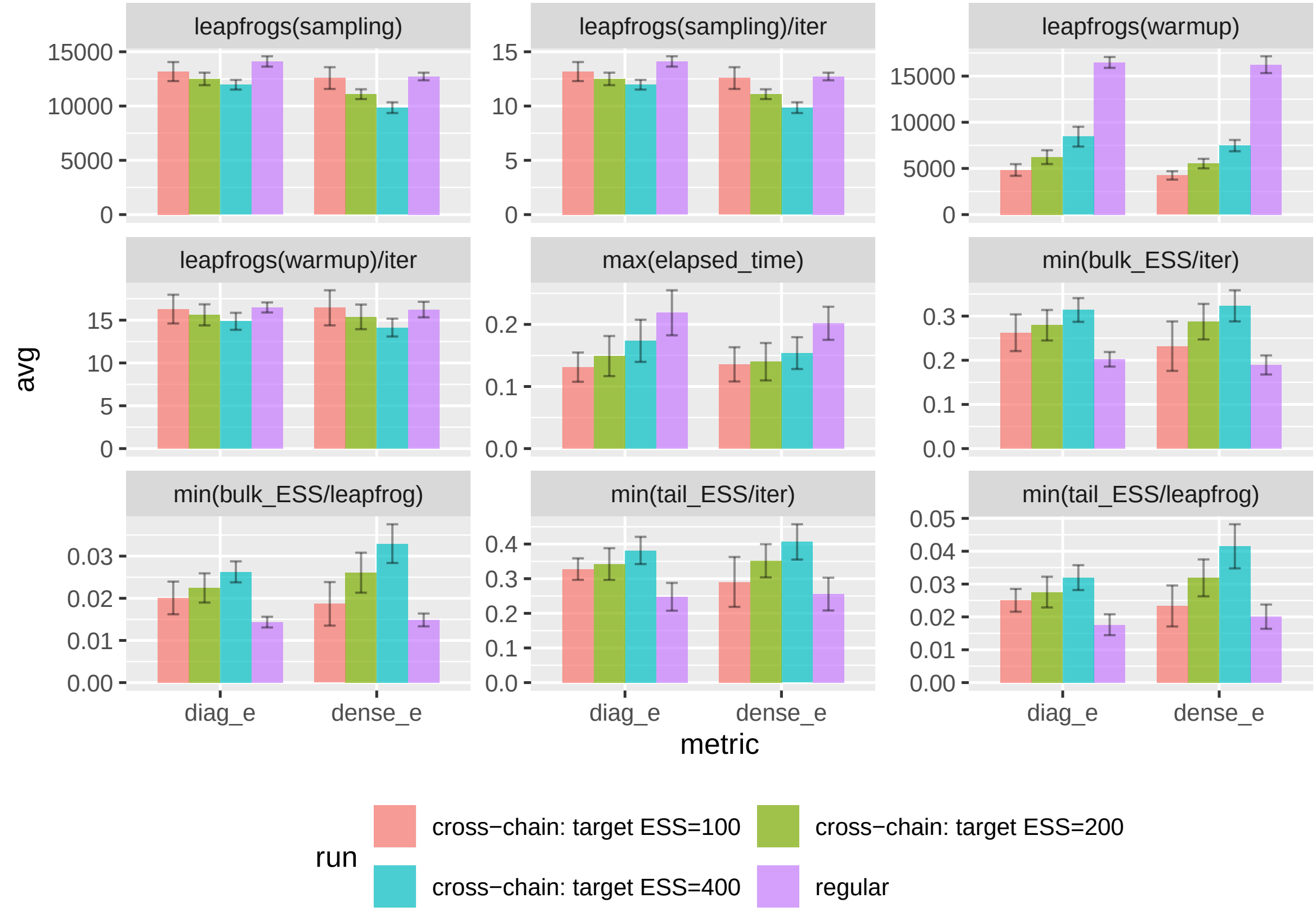

Figure 4: Cross-chain warmup performance: sblrc-blr model[3]

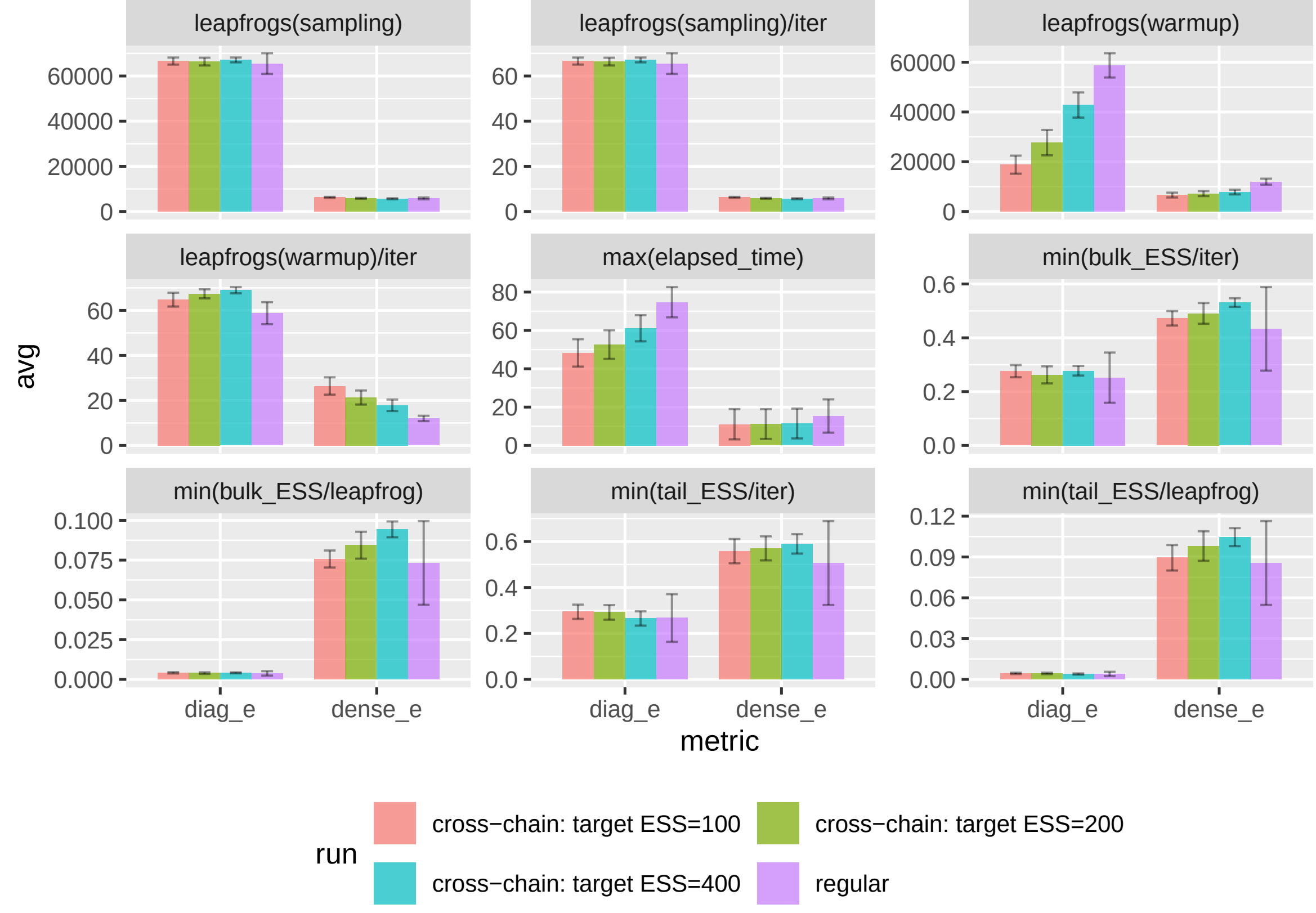

Figure 5: Cross-chain warmup performance: SIR model[3]

Multilevel parallelization: cross-chain warmup + within-chain parallelization

Method

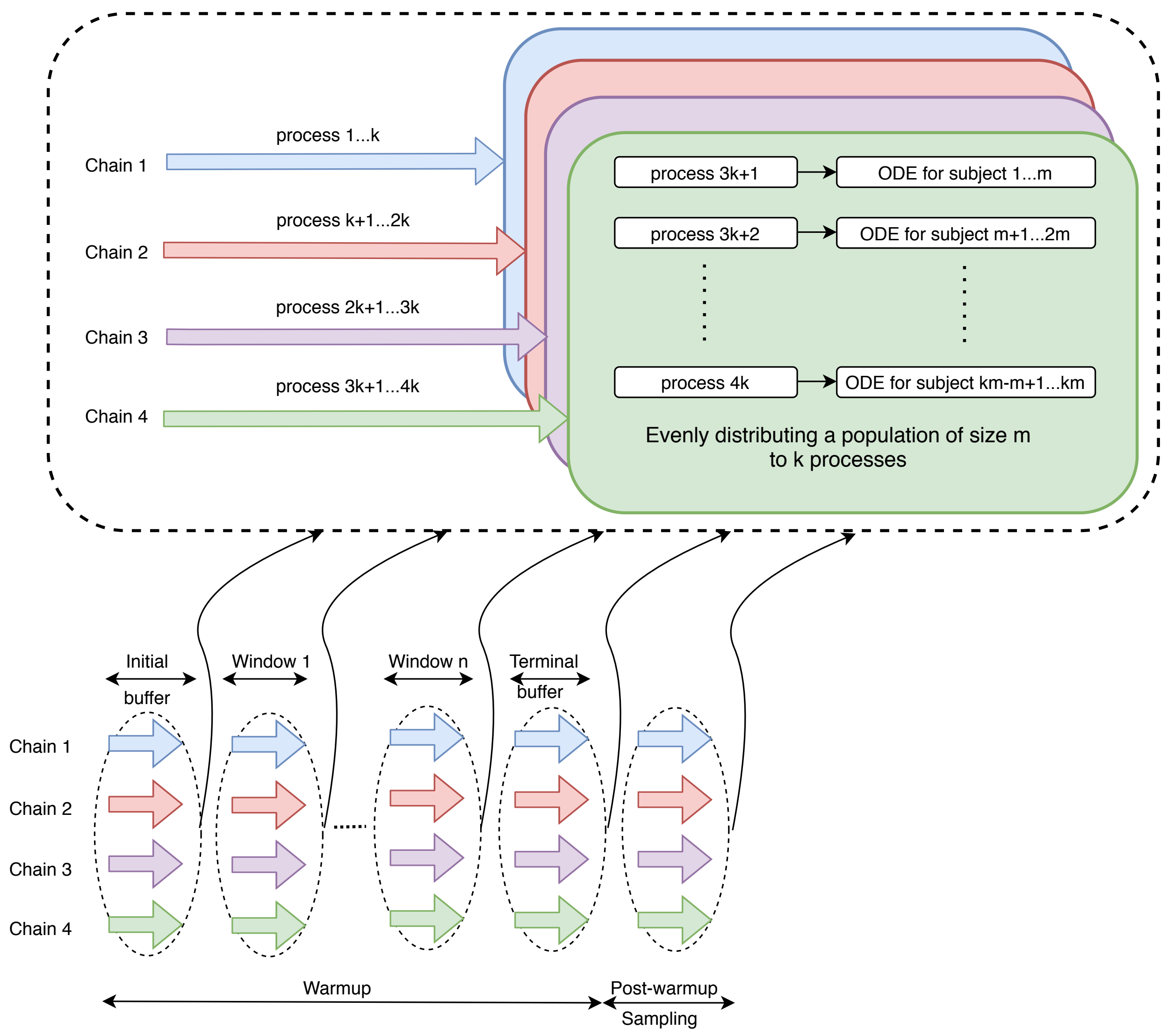

Figure 6: Multilevel parallelism for population models based on ordinary differential equations (ODE). A simplified version of Figure 1, the lower diagram shows the cross-chain warmup through multiple windows. In within-chain parallelization, as shown in the upper diagram, each chain has its own parameter samples (indicated by different colors), and dedicated processes for solving the population model. Thus the parallel computing is based on cross-chain level and within-chain level.

Level	Parallel operation	Parallel communication
1	Cross-chain warmup	At the end of warmup windows
2	Within-chain parallel ODE solver	During likelihhood evaluation

Table 1: A framework of *multilevel parallelism* for Bayesian inference of population models.

Example

We consider a time-to-event model for the time to the first grade 2+ peripheral neuropathy (PN) event in patients treated with an antibody-drug conjugate (ADC) delivering monomethyl auristatin E (MMAE). We call it Time-To-PN (TTPN) model, and analyze data using a simplified version of the model reported in [2]. We consider three treatment arms: fauxlatuzumab vedotin 1.2, 1.8 and 2.4 mg/kg IV boluses q3w x 6 doses, with 20 patients per treatment arm. In this model, each patient's PK is described by an effective compartment model (one-compartment), and PD by a linear model. The likelihood for time to first 2+ PN event is described by a hazard function that depends on the concentration effect through Weibull distribution. Two unknowns from PK model and the cumulative hazard form a three-component ODE system. Each evaluation of likelihood requires solving this 3-system for every patient.
ODEs corresponding to the entire population are solved by a single call of Torsten function `pmx_solve_group_rk45`. The three parameters of the Torsten model are:

- k_{e0} in effective compartment model.
- α the coefficient of linear PD model.
- β Weibull distribution scale parameter.

Warmup quality

Table 2 shows cross-chain and regular run performance (target ESS = 400). Consistent with the previous benchmark models, the cross-chain warmup reduces total run time without compromising ESS, leading to 15% wall time improvement.

	Cross-chain	Regular
`leapfrogs(warmup)`	`1.002e+04`	`1.588e+04`
`leapfrogs(sampling)`	`1.709e+04`	`1.831e+04`
`leapfrogs(warmup)/iter`	`1.822e+01`	`1.588e+01`
`leapfrogs(sampling)/iter`	`1.709e+01`	`1.831e+01`
`min(bulk_ESS/iter)`	`2.805e-01`	`2.340e-01`
`min(tail_ESS/iter)`	`3.482e-01`	`3.205e-01`
`min(bulk_ESS/leapfrog)`	`1.641e-02`	`1.277e-02`
`min(tail_ESS/leapfrog)`	`2.037e-02`	`1.749e-02`
`max(elapsed_time)`	`1.702e+03`	`1.979e+03`

Table 2: Cross-chain runs vs regular runs(target ESS=400)

Parallel speedup

Speedup is investigated by running the model with 4 chains using $n_{\text{proc}} = 4, 8, 16, 32, 60$ processes. Equivalently, there are 1, 2, 4, 8, 15 processes per chain for within-chain parallelization. With population size 60, this is also equivalent to having each process handle the solution of 60, 30, 15, 8, 4 subjects' ODE system, respectively.

- Both muiltilevel and within-chain-only parallel runs scale near-linearly up to 60 processes (15 processes per chain × 4 chains).
- In the range of $n_{\text{proc}} = 32, 60, 80$, multilevel runs exhibit a steady ~20% performance improvement, completely contributed by cross-chain warmup.

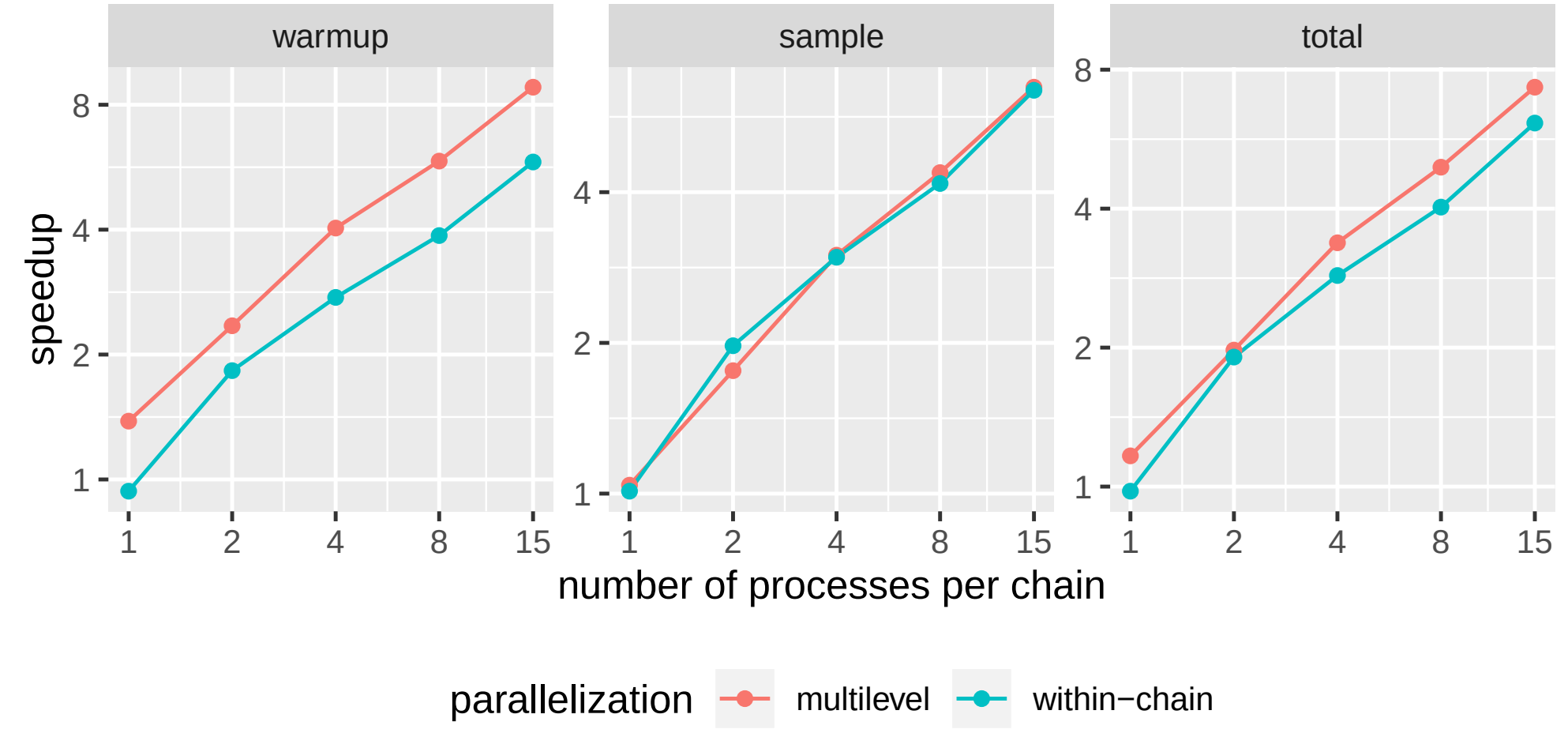

Figure 7: Multilevel scheme parallel performance: TTPN model (target ESS=400). Wall time speedup uses regular Stan run as reference. With all runs having 1000 post-warmup sampling iterations, in multilevel runs the number of warmup iterations is determinted at runtime, while both within-chain parallel runs and regular Stan runs have 1000 warmup iterations. Among 4 chains in a run, we use the one with maximum total walltime(in seconds) as performance measure, as in practice usually further model evaluation becomes accessible only after all chains finish.

References

[1] B. Carpenter et al., *Stan: A Probabilistic Programming Language*, Journal of Statistical Software, 76 (2017), pp. 1–32.

[2] D. Lu et al., *Time-to-Event Analysis of Polatuzumab Vedotin-Induced Peripheral Neuropathy to Assist in the Comparison of Clinical Dosing Regimens*, CPT: pharmacometrics & systems pharmacology, 6 (2017), pp. 401–408.

[3] M. Magnusson et al., *A posterior database (PDB) for bayesian inference*. `https://github.com/MansMeg/posteriordb`.

[4] Torsten Development Team, *Torsten: library of C++ functions that support applications of Stan in Pharmacometrics*. `https://github.com/metrumresearchgroup/Torsten`.

[5] A. Vehtari et al., *Rank-normalization, folding, and localization: An improved $\hat{R}$ for assessing convergence of MCMC*, arXiv:1903.08008 [stat], (2019). arXiv: 1903.08008.

[6] Y. Zhang and W. R. Gillespie, *Poster: Speed up ode-based modeling using torstens population solvers*, in StanCon 2019, Cambridge, UK, August 2019.

Conclusion

- Cross-chain warmup automates MCMC convergence assessment and adaptive transition to sampling. It also produces comparable or even better HMC tuning parameters. Benchmark shows the method is applicable to a wide range of models.
- Multilevel parallelism significantly improves population model inference performance.

See also

`github.com/metrumresearchgroup/acop_2020_torsten_parallelization`

Presented at ACOP11 Meeting; 9 -13 November, 2020. Copies available at: www.metrumrg.com/publications © Metrum Research Group 2020