Exact Resampling with Multiprocessing

L. Kekempanos, S. Maskell, J. Thiyagalingam, Y. Goulermas

UNIVERSITY OF LIVERPOOL

A fully distributed and parallel GENERIC particle filter is described which improves the computational complexity of the resampling algorithm. The sequential Monte Carlo methods, such as the GENERIC particle filter, are used in a variety of applications including but not limited to finance and tracking.

EPSRC
Engineering and Physical Sciences Research Council

Components & Complexity

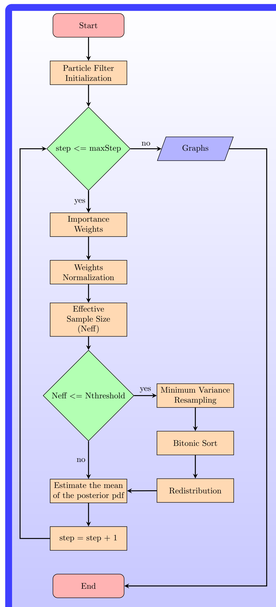

Figure 1: Parallel GENERIC particle filter flowchart.

The GENERIC particle filter (fig. 1, 2), belongs to a wider class of algorithms called sequential Monte Carlo methods (SMC methods) [1]. The main idea of the SMC methods is to estimate key characteristics of a model using a number of particles. The GENERIC particle filter can be described with a set of algorithmic components. To construct the parallel GENERIC particle filter the proposed strategy is to start building simple components, such as the parallel cumulative summative summation, the differences between adjacent elements, weights normalization and proceed to more advanced algorithms, such as the bitonic sort and redistribute. The correct positioning of the components lead to the final algorithm. The computational complexity of each algorithmic component is provided in the following table (fig. 3).

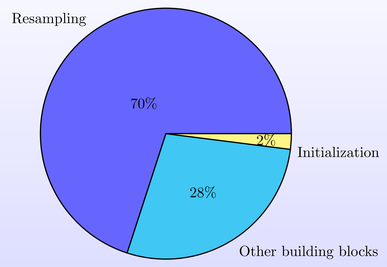

Figure 2: Computational effort for the serial GENERIC particle filter.

	Sequential	Parallel	
Algorithmic Component	Time	Time	Space
Summation	$O(N)$	$O(log_2 N)$	$O(N)$
Cumulative Summation	$O(N)$	$O(log_2 N)$	$O(N)$
Differences Between Adjacent Elements	$O(N)$	$O(1)$	$O(N)$
Max/Min Value	$O(N)$	$O(log_2 N)$	$O(N)$
Weights Normalization	$O(N)$	$O(log_2 N)$	$O(N)$
Minimum Variance Resampling	$O(N)$	$O(log_2 N)$	$O(N)$
Sort	$O(Nlog_2 N)$	$O((log_2 N)^2)$	$O(N)$
Redistribution	$O(N)$	$O((log_2 N)^2)$	$O(N)$

Figure 3: Time/space complexity for each component. The input size, N, is a number of a power of 2.

Importance of the Resampling Algorithm

The resampling algorithm [4],[5] is a treatment to a problem called "weight degeneracy" (particles have negligible weights and large variance, which leads to inaccurate estimation). The comparison between different SMC methods is formally achieved with evaluation methods, such as the root mean square error (RMSE, fig. 5). A visual comparison (fig. 4) of two different SMC methods, the sequential importance sampling (SIS; it never uses the resampling algorithm) and the sequential importance resampling (SIR; it uses the resampling algorithm in every time step), for a given model [1], shows that the SIR is a better estimator than the SIS. This further underpins the importance of parallelizing the resampling algorithm.

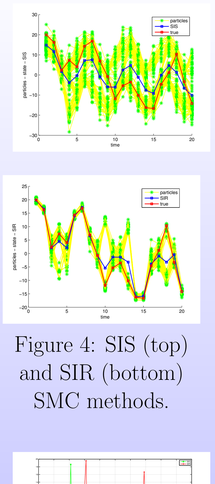

Figure 4: SIS (top) and SIR (bottom) SMC methods.

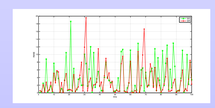

Figure 5: RMSE comparison of the SIS and the SIR methods.

Bitonic Sort

In 1968 Kenneth Batcher invented one of the fastest sorting algorithms, the bitonic sort (fig. 6, 7). Fifteen years later, in 1983, Ajtai-Komlós-Szemerdi proposed the AKS sorting network. The AKS sorting network is theoretically faster than the bitonic sort, but practically it has been proven that this is only true when the input size, N, is is approximately more than 10^{21} [3].The bitonic sort requires two phases; The construction and the sorting of the bitonic sequence.

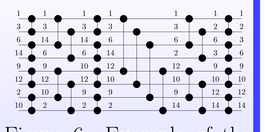

Figure 6: Example of the bitonic sort with 8 numbers.

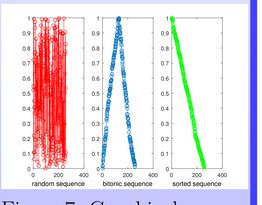

Figure 7: Graphical presentation using 256 numbers.

Redistribute

The redistribute algorithm is a divide and conquer algorithm, which receives two input vectors, the old population of particles and the number of copies. Every element of the second vector describes the number of copies each element of the first vector needs to be copied in order to produce the new population of particles. A global configuration of the algorithm is that the second vector for every node begin with non-zero values (fig. 9). An example of the algorithm is provided in (fig. 8).

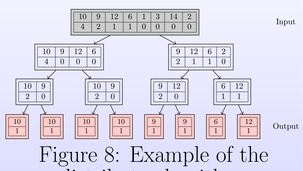

Figure 8: Example of the redistribute algorithm.

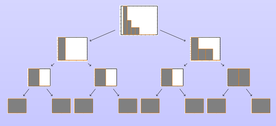

Figure 9: Histogram of the number of copies.

Results

The Apache Spark [7] framework is used in a single computer. The algorithm gives a performance improvement, but not what exactly expected (fig. 10). Given an extra source the expectation is to have an equal improvement in the computation time (Amdahl's law). The update of a single value using a "heavy" computation multiple times, leads to speedup closer to the "ideal" value (fig. 11). This implies that if the operation is very simple in terms of computation then the communication dominates the computation time.

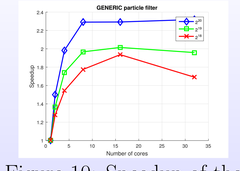

Figure 10: Speedup of the GENERIC particle filter.

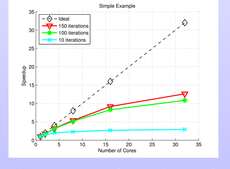

Figure 11: Speedup of single map update.

References

[1] S. Arulampalam, S. Maskell, N, Gordon and T. Clapp, *A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking*, 2002.

[2] S. Arulampalam, S. Maskell, B. Alun-Jones and M. Macleod, *A Single Instruction Multiple Data Particle Filter*, 2006.

[3] L. Natvig, *Logarithmic time cost optimal parallel sorting is not yet fast in practice*, 1996.

[4] J. Hol, T. Schon, F. Gustafsson, *On Resampling Algorithms For Particle Filters*, 2006.

[5] T. Li, M. Bolic, and P. Djuric, *Resampling Methods for Particle Filtering*, 2015.

[6] S. Sutharsan, T. Kirubarajan, T. Lang, M. McDonald, *An Optimization-Based Parallel Filter for Multitarget Tracking*, 2012.

[7] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, I. Stoica, *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*, 2012.