Title

Distributed Memory Parallel Markov Random Fields Using Graph Partitioning

Authors

Heinemann, C
Perciano, T
Ushizima, D
et al.

Publication Date

2017-12-01

DOI

10.1109/bigdata.2017.8258318

Peer reviewed

Distributed Memory Parallel Markov Random Fields using Graph Partitioning

C. Heinemann[1], T. Perciano[1,3], D. Ushizima[1,2,3], E. W. Bethel[1]
[1]*Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[2]*University of California, Berkeley, CA, USA*
[3]*Center for Advanced Mathematics for Energy Research Applications (CAMERA), LBNL, Berkeley, CA, USA*
Email: cheinemann@lbl.gov, tperciano@lbl.gov, dushizima@lbl.gov, ewbethel@lbl.gov

***Abstract*—Markov random fields (MRF) based algorithms have attracted a large amount of interest in image analysis due to their ability to exploit contextual information about data. Image data generated by experimental facilities, though, continues to grow larger and more complex, making it more difficult to analyze in a reasonable amount of time. Applying image processing algorithms to large datasets requires alternative approaches to circumvent performance problems. Aiming to provide scientists with a new tool to recover valuable information from such datasets, we developed a general purpose distributed memory parallel MRF-based image analysis framework (MPI-PMRF). MPI-PMRF overcomes performance and memory limitations by distributing data and computations across processors. The proposed approach was successfully tested with synthetic and experimental datasets. Additionally, the performance of the MPI-PMRF framework is analyzed through a detailed scalability study. We show that a performance increase is obtained while maintaining an accuracy of the segmentation results higher than 98%. The contributions of this paper are: (a) development of a distributed memory MRF framework; (b) measurement of the performance increase of the proposed approach; (c) verification of segmentation accuracy in both synthetic and experimental, real-world datasets.**

***Keywords*-Markov Random Fields; image segmentation; parallel parameter estimation/optimization; distributed memory; material science**

I. Introduction

Applying image processing algorithms to the field of material science is important for studying the properties of material samples, analyzing global structure (e.g., porosity) and microstructures (e.g., fibers, grains, and pores), etc. [1] [2] [3]. Material science technology is becoming more advanced and cameras now provide greater amounts of detail regarding microscopic structures in resulting datasets. Thus, the datasets are larger and contain more data to analyze. Analysis runtime is increased as a result. Image processing techniques can be used to assist in the analysis of such datasets. Many current approaches, which are single socket in nature, are unable to accommodate growing data sizes due to single-node memory limitations. Consequently, improving current techniques through a scalable, general purpose image analysis framework becomes increasingly valuable.

A. Background

Given large data rates and sizes, machine learning for data acquisition [4], as well as for the automation of feature detection and extraction, represents key steps in reducing data while gaining insight from image structures. Numerous categories of automated feature extraction exist [5] [6], but many are reliant on image segmentation techniques supported by unsupervised learning [7]. Broadly used to analyze experimental data, unsupervised segmentation algorithms [8] enable the grouping of picture elements or, in essence, collecting tokens that "belong together". Such algorithms can gather meaningful groups of information by searching for hidden structures in unlabeled data. Advantages include dispensable training data and potential for data reduction such as removal of non-contributing portions of images including background and artifacts.

Unsupervised image segmentation can be done with the help of a probabilistic graphical model known as Markov random fields (MRF) [1]. MRFs represent discrete data by modeling neighborhood relationships. Moreover, such models provide ways to formulate theoretical algorithms for the processing of functions related to graphs. These can then be used to solve problems in image processing such as segmentation/classification, image registration, feature detection, and texture analysis [9]–[14]. The idea behind MRF models is to use undirected graphical models to obtain a higher level representation of an image. In doing so, these models consolidate structure representation for later image processing and analysis [15]. In fact, many different concepts can be combined using graphs in pursuance of modeling real-world problems.

B. Performance of Markov random fields methods

Despite the several advantages of using MRFs, the algorithms are limited by performance when running large and complex datasets in serial, taking excessive amounts of time to generate valuable results. Strategies

have been proposed, such as graph-cut techniques, to overcome the computational complexity of MRF optimization (NP-hard) and provide a powerful alternative from both computational and theoretical viewpoints. However, they are often restricted to specific types of models (first-order MRFs) [16] and energy functions (regular or submodular) [16].

For higher-order MRFs and non-submodular functions, some strategies using parallelized graph cuts and parallelized Belief Propagation have also been proposed [17]–[20], but these approaches typically depend on orderly reduction or submodular functions [21]. These are undesirable constraints when dealing with large and complex image datasets because they limit the contextual modeling of the problem. Problems with such datasets require higher-order potentials. These potentials provide more accurate characterization and richer interactions among random variables. This, consequently, impacts the definition of image priors and image analysis results.

In order to circumvent such drawbacks, recent works [22] [23] have proposed theoretical foundations for distributed parameter estimation in MRF. These approaches make use of a composite likelihood, allowing the distribution of sub-problems across processors to be solved in parallel. Under general conditions on the composite likelihood factorizations, the distributed estimators are proven to be consistent. The Linear and Parallel (LAP) [24] algorithm parallelizes naturally over cliques and, for graphs of bounded degree, its complexity is linear in the number of cliques. It is fully parallel and, for log-linear models, it is also data efficient. To estimate parameters, it requires only the local statistics of the data, i.e., considering only pixel values of local neighborhoods.

Perciano *et al.* [25] proposed a graph-based model, referred to as Parallel Markov random fields (PMRF), that exploits MRFs to segment images. Both the optimization and parameter estimation processes are parallelized using the LAP method [24]. Even though the proposed multi-threaded approach highly improves the computational performance of the MRF-based segmentation algorithm, its shared memory parallelism is a limit to its scalability. Therefore, it is necessary to reconfigure PMRF for distributed memory parallelism. The work presented here overcomes these previous limitations by reformulating the algorithm for use on a distributed memory platform, which can accommodate larger problem sizes and faster computations.

Aiming to overcome previous limitations of existing algorithms, we propose a distributed memory approach to PMRF (MPI-PMRF). The goal of utilizing distributed memory parallelization, specifically the Message Passing Interface (MPI), with the PMRF algorithm is to increase performance by running multiple subsets of the dataset simultaneously. Because MPI is widely accepted and known, as well as generally being readily available, MPI is an optimal choice for expanding the scope of the work distribution from the threaded PMRF version on a single node to the MPI version that can be scaled across multiple nodes. In doing so, we are delivering a valuable tool that can be used to analyze large and complex experimental datasets. Redesigning the PMRF algorithm to take advantage of distributed memory parallelization provides the ability to analyze increasingly large datasets and to obtain accurate results in a more reasonable amount of time.

II. Design and Implementation

MPI-PMRF is developed based on two main building blocks: a statistical MRF graph model and an optimization approach using a graph partitioning strategy. The MRF graph model used for MPI-PMRF is taken from [25] and is described in further detail in Section I-B. In the following sections, we provide a detailed explanation of the two main components.

A. Markov random field model

In a MRF model, the optimization process uses a global energy function to find the best solution to a similarity problem, such as the best pixel space partitioning or the best matching. The energy function consists of a data term and a smoothness term. For image segmentation, we use the mean of the intensity values of a region as the data term. The smoothness term takes into account similarities between regions. The goal is to find the best labeling for the regions so that the similarity between two regions with the same label is optimal [26].

Given an image represented by $\mathbf{y} = (y_1, \ldots, y_N)$, where each y_i is a region, we seek a configuration of labels $\mathbf{x} = (x_1, \ldots, x_N)$ where $x_i \in L$ and L is the set of all possible labels, $L = \{0, 1, 2, \ldots, K\}$. The MAP criterion [1] states that it is necessary to find a labeling $\mathbf{x}^*$ that satisfies $\mathbf{x}^* = \underset{x}{\operatorname{argmax}}\{P(\mathbf{y}|\mathbf{x}, \Theta)P(\mathbf{x})\}$, which can be rewritten in terms of the energies as $\mathbf{x}^* = \underset{x}{\operatorname{argmin}}\{U(\mathbf{y}|\mathbf{x}, \Theta) + U(\mathbf{x})\}$, where $U(\mathbf{y}|\mathbf{x}, \Theta)$ is the conditional energy function and $U(\mathbf{x})$ is the prior energy function, based on the sum of the prior and joint probabilities. The prior probability $P(\mathbf{x})$ is a Gibbs distribution, and the joint probability distribution is $P(\mathbf{y}|\mathbf{x}, \Theta) = \prod_i P(y_i|\mathbf{x}, \Theta) = \prod_i P(y_i|x_i, \theta_{x_i})$, where $P(y_i|x_i, \theta_{x_i})$ is a Gaussian distribution with parameters $\theta_{x_i} = (\mu_{x_i}, \sigma_{x_i})$ and $\Theta = \{\theta_l | l \in L\}$ is the parameter set. These parameters are estimated during the optimization process and are used to find the optimal labeling for the problem.

B. MPI-PMRF algorithm

MPI-PMRF is a refactorization and extension of the previous work [25] that is capable of running on distributed memory platforms by reconfiguring the work distribution and calculation of the optimal solution to take advantage of distributed memory parallelization. Combining graph partitioning with the MRF provides the ability to run the optimization process in a distributed memory fashion with MPI-PMRF and, therefore, find the optimal labeling for the problem faster. The proposed MPI-PMRF algorithm consists of the following three basic steps:

1) Given an original image and an oversegmention of that image (higher level partitioning of the image into regions), construct a graph model;
2) Refactor the graph model according to the Markov random fields model to target the image segmentation;
3) A distributed approach is used for the optimization process to obtain the optimal labeling for the graph.

A detailed description of the algorithm is presented in Algorithm 1.

Algorithm 1 Distributed memory version of the Markov Random Field algorithm using graph partitioning and parameter estimation (MPI-PMRF). Line 8 is run in parallel to distribute the largest amount of work to increase performance.

Input: Original image, oversegmentation, number of classes
Output: Segmented image and estimated parameters
1: $K \leftarrow$ number of classes
2: Initialize parameters and initial labels randomly
3: Create graph from oversegmentation
4: **for** each Expectation Maximization iteration **do**
5: Divide graph into subgraphs (cliques) based on number of MPI processes to be used
6: Distribute cliques to MPI processes
7: **for** each non-zero clique of the graph **do**
8: Run Expectation Maximization and Maximum a Posteriori optimizations on assigned MPI processes
9: **end for**
10: Gather parameter estimation information for subgraphs
11: Update parameters
12: **end for**
13: Generate resulting output image

Input to the MPI-PMRF is the original image, as well as an oversegmented image. The oversegmentation is a fast but low-accuracy approach for image segmentation [25]. Pixels from the input image with similar characteristics are grouped into regions. Line 3 of Algorithm 1 shows such regions being used for graph creation. Second, the resulting regions make up the nodes of the graph representation of the image. The graph partitioning process is done using the Linear and Parallel (LAP) algorithm [24], which creates subproblems for each maximal clique in the problem and allows simultaneous parameter estimation and optimization to be executed for every generated subgraph of the original graph [24] [25].

The MPI-PMRF algorithm divides the full parameter estimation process into several fully independent subproblems, as seen in Line 5 of Algorithm 1. See Section II-A for more detail. The natural parallelization capabilities of these sub-problems, or cliques, makes them an ideal candidate to be solved in parallel. Line 6 of Algorithm 1 shows that the framework utilizes MPI to distribute the optimization of the cliques, which is the largest amount of work in the algorithm, to the allocated processes in order to increase performance. Line 8 of Algorithm 1 shows the optimization processes that are performed in parallel.

When clique optimization is completed, all results are combined by reassociating the regions of the graph with the region's respective pixels from the original input image. Upon completion of the conversion from regions to pixels, the final output image is written, given in Line 13 of Algorithm 1.

III. Results

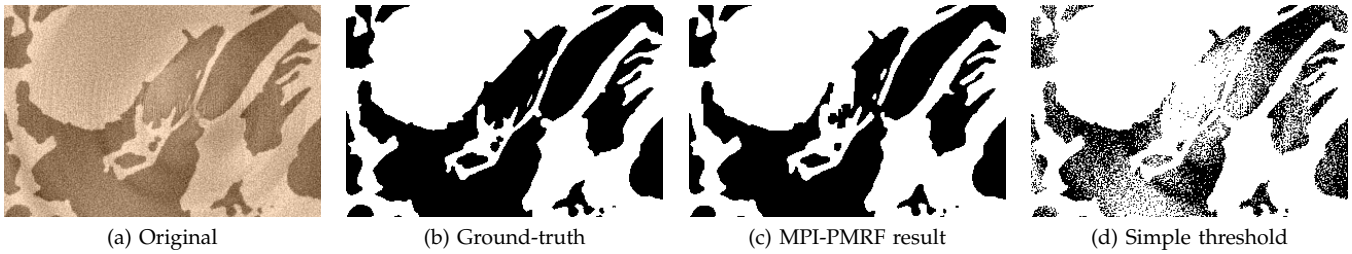
In this section, we describe the methodology used to evaluate the proposed approach in terms of both segmentation precision and performance.

A. Image segmentation experiments

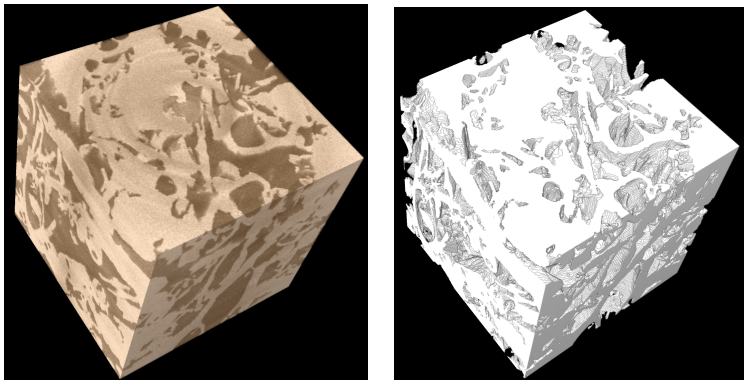
1) Datasets: MPI-PMRF is tested with two different datasets. First, we use a synthetic dataset to verify the correctness and accuracy of the MPI-PMRF framework. Then, we use an experimental dataset to show how MPI-PMRF performs when analyzing real-world material science problems.

We selected the synthetic dataset from the 3D benchmark made available by the Network Generation Comparison Forum (NGCF) [1]. The NGCF datasets are a global, recognized standard and contain a known ground-truth, supporting the goal of guaranteeing reproducibility of accurate results with MPI-PMRF. The full synthetic dataset is 268 MB and consists of 512 image slices of dimensions 512×512. For the purposes of this analysis, the original stack is corrupted by noise (salt-and-pepper) and additive Gaussian with $\sigma = 100$. Additionally, ringing artifacts are also simulated in the sample to closer resemble real-world results. This dataset emulates a very porous fossiliferous outcrop carbonate - Mt. Gambier limestone - from South Australia. Mimicking properties of real-world datasets provides the verification that MPI-PMRF will produce

[1] http://people.physics.anu.edu.au/~aps110/network_comparison



(a) Original (b) Ground-truth (c) MPI-PMRF result (d) Simple threshold



(e) 3D rendering of the original data (f) 3D rendering of the MPI-PMRF result

Figure 1: Results after applying MPI-PMRF to the synthetic dataset. (a) Original region of interest from the noisy data; (b) Ground-truth; (c) Result obtained by the proposed MPI-PMRF; (d) Result obtained using a simple threshold; (e) 3D rendering of the original noisy dataset; (f) 3D rendering of the result obtained by MPI-PMRF.

correct results with datasets where the ground-truth cannot be tested.

The experimental dataset contains cross-sections of a geological sample and uses a gray scale value to represent information regarding the x-ray attenuation and density of the scanned material. Isotropy and pixel resolution at micrometer scale, as well as the possibility of creating 3D virtual models, make microCT [27] imaging advantageous for material analysis. Using synchrotron light radiation to probe the material structure, the LBL Advanced Light Source X-ray beamline 8.3.2 [28] scans samples, which are subjected to energies between 10 and 45keV, with a 1% bandpass, CCD camera Cooke PCO 4000, Kodak chip with 4008×2672 pixels, 14-bit, 9 micron square pixels. As part of the normal data processing pipeline, when the data comes off the instrument, it is run through a reconstruction algorithm that processes a 3D image that is 3 GB in size, consisting of 500 image slices with dimensions of 1813×1830.

While the synthetic and experimental datasets, which are 268 MB and 3 GB in size, respectively, are not particularly "large", we limit our scalability studies, presented in Section III-B, to these two datasets due to the fact that we need to be able to run in a serial configuration for the purposes of comparison with the parallel version of the algorithm.

2) Evaluation metrics: In order to determine the precision of the segmentation results we use the metrics *precision*, *recall*, and *accuracy* defined as follows:

$$precision = \frac{TP}{TP + FP} \quad (1)$$

$$recall = \frac{TP}{TP + FN} \quad (2)$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (3)$$

where TP stands for True Positives, TN for True Negatives, FP for False Positives, and FN for False Negatives.

In addition, we also calculate the porosity (ratio between void space and total volume), or ρ, which is given by

$$\rho = \frac{V_v}{V_t}, \quad (4)$$

where V_v is the volume of the void space and V_t is the total volume of the void space and solid material combined.

3) Results: We apply the MPI-PMRF framework to the two datasets described previously by processing each individual 2D cross-section of the 3D volumes. The output should ideally separate the solid phase

from the void phase to show structures such as cavities and granularity of the material.

Figure 1 presents the results for the synthetic dataset. The ground-truth (b) and the result obtained using the proposed method (c) are very similar, indicating a good convergence of the algorithm. We also present the result obtained using a simple threshold (d), showing the inability of such an approach to successfully separate the two different phases. For this dataset, the evaluation metrics values are 99.52% for *precision*, 98.69% for *recall*, and 98.99% for *accuracy*. The result of porosity for the synthetic dataset when utilizing the MPI-PMRF is 44.06%, which is very close to the ground truth value of 43.6%.

Figure 2 presents the results when running the experimental dataset with MPI-PMRF. The algorithm successfully separates the rock-structures from the background (b). Using a simple threshold results in different output (c). Given that a ground-truth is not available for the experimental dataset, we are not able to calculate the evaluation metrics; however, the segmentation results are identical to those presented in [25]. The porosity is calculated at 42.32%.

Using the MPI-PMRF framework to obtain accurate segmentation results is extremely valuable to material scientists. Segmenting geological samples aids in the understanding of, for example, carbon sequestration – geologic storage of captured CO_2 in underground rock formations. Precise image segmentation provides valuable measurements of porosity and permeability of these samples. The identification of the different phases in the image makes possible the assessment of the material's 3D architecture and the measurement of the structures involved.

B. Performance analysis

1) Methodology: To better understand the scalability of MPI-PMRF, we conducted a performance student on 1, 2, 4, 12, 24, 48, 96, 192, 384, and 768 processes, specifically targeting the framework's optimization process due to the fact that it takes the largest amount of time to complete and it is designed to be parallelized. Leveraging on the parallelization capabilities of the optimization process provides the ability to run larger and more complex datasets in a reasonable amount of time. The results show that, when running in parallel, it is possible to achieve a performance increase. The performance increase obtained at varying levels of concurrency is discussed in detail in Sections III-B3 and III-B4.

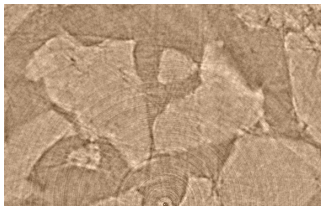
2) Platforms: MPI-PMRF was developed and tested on the Edison supercomputer at the National Energy Research Scientific Computing Center (NERSC). The complete Edison system is a Cray XC30 system with a peak performance of 2.57 PFLOPS (10^{15} floating point operations per second), 133,824 compute cores, 357 terabytes of memory, and 7.56 petabytes of disk [29]. In addition, both Edison and Edison's network are user-based systems, potentially leading to a fluctuation in result times. Additional details regarding results can be found in Sections III-B3 and III-B4.

3) Analysis of Scalability: Our approach for understanding scalability is to measure runtime performance, in seconds, at varying levels of concurrency for the different datasets. We accommodate variability by running each experiment configuration 5 times and averaging the results. Additional performance metrics are also analyzed in Section III-B4. Every experiment was performed 5 times and the average runtime was calculated to account for any possible system variability on Edison, as stated in Section III-B2. Figure 5 provides the numeric averages for both datasets at varying concurrencies.

Figures 3 and 4 show the runtime performance of the proposed approach on the datasets in a logarithmic fashion to better illustrate the performance increase obtained. In both figures, the vertical axis represents time, in seconds, and the horizontal axis is the image slice of the dataset. The individual colored lines each represent a different concurrency. Because every experiment was run 5 times, each given line represents the average of every run for the given slice at the respective concurrency. The gray areas surrounding the colored lines represent the standard deviation from the average shown. As concurrency increases, we observe that the standard deviation of runtime decreases. The standard deviation decreases with increasing concurrency because it is relative to the average; if the average is higher (representing a larger runtime), as it is at lower concurrencies, the standard deviation will be larger.

In both Figures 3 and 4, one can observe that there is variation in runtime between image slices, particularly at the lower concurrencies. Such a behavior is expected due to the fact that, since the runtime of the optimization process is proportional to the number of cliques run by each process, the runtime becomes a function of the complexity of the underlying data on which MPI-PMRF is applied.

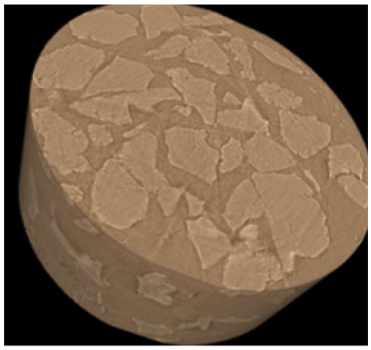
An important factor to note is the difference between the results for the datasets. The synthetic dataset does not contain as large of a variation in the number of cliques as the experimental dataset. Because the experimental dataset is a real-world problem, it is subject to outside factors such as noise or outside anomalies that are not observed in a synthetic dataset. As stated in Section III-A1, the results for the synthetic dataset are used as a verification technique while the results for



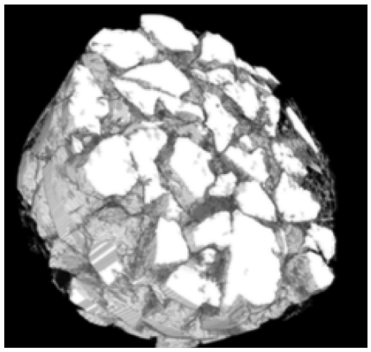
(a) Original

(b) MPI-PMRF result

(c) Simple threshold



(d) 3D rendering of the original data



(e) 3D rendering of the MPI-PMRF result

Figure 2: Results applying MPI-PMRF to the experimental dataset. (a) Region of interest from the original data; (b) Result obtained by the proposed MPI-PMRF; (c) Result obtained using a simple threshold; (d) 3D rendering of the original dataset; (e) 3D rendering of the result obtained by MPI-PMRF.

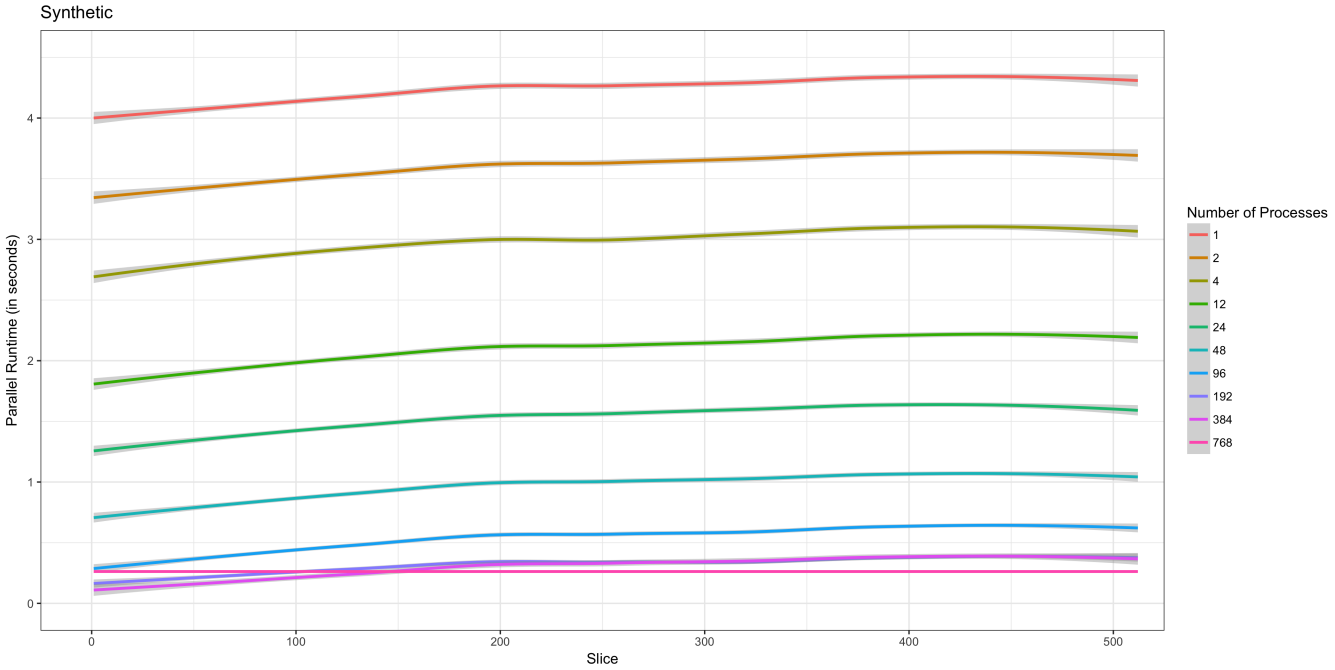


Figure 3: When measuring runtime (in seconds) of the synthetic dataset, the results show the overall decrease in runtime as concurrency increases. See Section III-B3 for more detail.

the experimental dataset are to present an application of MPI-PMRF.

Also, the runtime for the optimization process is directly proportional to the number of cliques processed for a given slice of the dataset. The runtime inherently reflects the complexity of the clique being

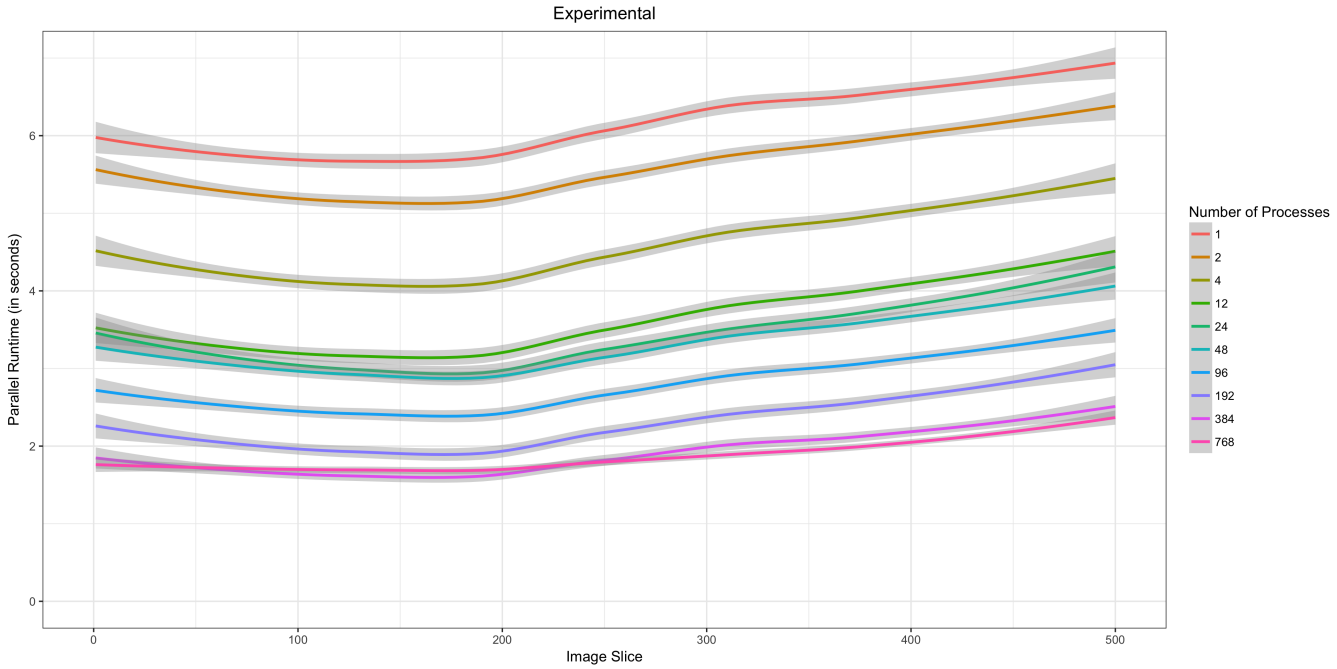


Figure 4: When measuring runtime (in seconds) of the experimental dataset, the results show the overall decrease in runtime as concurrency increases. See Section III-B3 for more detail.

Number Processes	Synthetic Time (in seconds)	Experimental Time (in seconds)
1	70.6	633.9
2	37.4	340.4
4	20.2	129.9
12	8.3	50.3
24	4.7	38.4
48	2.7	33.5
96	1.7	19.1
192	1.4	11.9
384	1.4	7.6
768	1.3	6.9

Figure 5: Average time for the optimization process to execute on given number of processes.

optimized. Additionally, the distribution of the work used in the experiments is important to note as well. The number of cliques of the graph is divided by the number of processes on which the optimization process will run. Each process receives an equal subsection of cliques while the final process receives a equal sized subsection plus the remainder of cliques that did not divide evenly. While it is possible that the number of cliques run by each process could be equal, the number of individual nodes within the cliques run on each process can be drastically different due to the fact that not all cliques are composed of the same number of nodes. We discuss this subject further in Section III-B4.

An important observation to point out is that, particularly in the results for the synthetic dataset in Figure 3, there is very little, if any, increase in performance at the highest concurrencies (192, 384, and 768). In fact, there are times when the lower concurrencies actually performance better than the higher concurrencies. When looking at the results for the experimental dataset in Figure 4, it can be seen that there is a similar outcome, but not to the same extent. There is more of a performance increase in the experimental dataset at the higher concurrencies than in the synthetic dataset at the same concurrencies. Because the synthetic dataset is much smaller than the experimental dataset, it can be concluded that there simply is not enough data in the synthetic dataset to be divided among such large numbers of processes. While the synthetic dataset is useful in verifying the validity of the MPI-PMRF framework, it does not scale as well as the larger, more complex real-world datasets. Future work will explore this idea in more detail, to examine scalability on datasets much too large to fit within memory on a single node. Such work will be valuable to experimental scientists faced with an increasing deluge of data.

4) Performance metrics: In this section, we look more deeply into runtime performance to better understand scalability and efficiency. Moreland and Oldfield [30]

present the idea that traditional performance metrics do not perform as well for large-scale studies with data that cannot run on a single serial computer. They define the concepts of rate and efficiency to simplify scalability studying by removing the dependency between the problem size and the number of processing elements. Such metrics, in general, expect that the problem has a distributed workload balance. MPI-PMRF does not exhibit such a characteristic and this is reflected in the results.

The definitions of efficiency and rate are given below. n is input size, p is number of processing elements, T is time, C^* is the cost of running the algorithm in serial (minimal cost), and C is the cost when running on a given number of processes.

1) **Efficiency** is the ratio of the best minimal cost, or the cost of running in serial, to the actual cost of running on numerous processes and is given by

$$E(n,p) = \frac{C^*(n)}{C(n,p)} \tag{5}$$

2) **Rate** is calculated in terms of size of input computed per unit time rather than absolute run time and is given by

$$R(n,p) = \frac{n}{T(n,p)} \tag{6}$$

Figure 6 shows the results for the synthetic dataset's efficiency and rate. Figure 7 shows the results for the experimental dataset's efficiency and rate. The ideal value for efficiency is a constant value of 1, represented by the solid black line in Figures 6a and 7a; the ideal slope of the rate is linear, represented by the solid black line in Figures 6b and 7b. The efficiency and rate of MPI-PMRF do not follow the ideal trends, though. As stated previously, the poor results for efficiency and rate are due to the fact that the workload distribution is not balanced across the processes executing the optimization.

Two factors contribute to the imbalance that causes poor scalability. The first factor is the number of cliques itself. During workload distribution, it is not guaranteed that the total number of cliques will divide evenly among the allocated processes. Consequently, not all processes receive an equal amount of work. This indicates that some processes will take longer to execute and, thus, have larger runtimes. The second factor is the imbalance within the cliques themselves. While we may be able to evenly distribute cliques across processes, because the amount of computation per clique varies as a function of clique complexity, load imbalance may result when processes having more complex cliques have more work to do than processes having less complex cliques.

As concurrency increases, we see performance degradation due to the load imbalance caused by different processes having different amounts of work. When running at higher concurrencies, the unbalanced workload leads to a drop in efficiency. Similarly, we see that there is a decrease in the rate at which the performance increases. Figures 6 and 7 show where the results do not follow the ideal trends. While the rate of performance increase does not follow the ideal trend, we see that the MPI-PMRF framework still provides a small performance increase when running at higher concurrencies, making it advantageous when analyzing large datasets as opposed to serialized analysis.

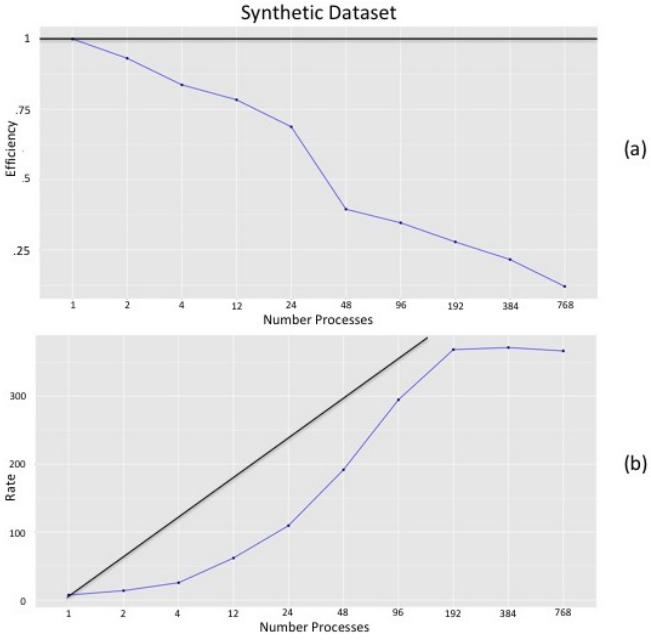


Figure 6: (a) The efficiency of the synthetic dataset does not follow the ideal efficiency, which is equal to 1, but still provides an increase in performance. (b) The rate of the synthetic dataset does not exactly follow the ideal rate, with a slope equal to 1, but still provides a performance increase at different concurrencies and executes faster than when running in serial. See Section III-B4 for more detail.

IV. Conclusions

The MPI-PMRF framework is a distributed memory parallel version of an advanced image segmentation based on Markov random fields optimization. Equipment in material science is becoming more advanced and resulting in larger datasets, making MPI-PMRF a valuable tool to help scientists with the analysis of such datasets. In fact, our framework can potentially help scientists discover valuable information from the

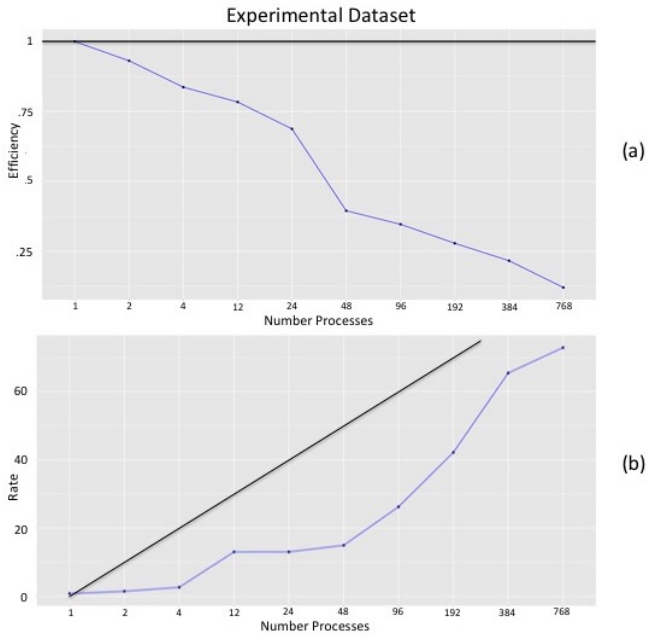


Figure 7: (a) The efficiency of the experimental dataset does not follow the ideal efficiency, which is equal to 1, but still provides an increase in performance. (b) The rate of the experimental dataset does not exactly follow the ideal rate, with a slope equal to 1, but still provides a performance increase at different concurrencies and executes faster than when running in serial. See Section III-B4 for more detail.

datasets by minimizing the impact on performance given its parallel nature.

The performance analysis and scaling studies conducted in this work show runtime performance gains as we increase concurrency, which is helpful for the experimental scientists who need to make use of such methods. Our studies also show performance limitations at higher concurrencies that are due to modest load imbalance. The load imbalance is due to the fact that, while we may be able to distribute cliques nearly evenly across processes, the per-clique workload is a function of clique complexity.

Future work will examine more advanced methods of workload estimation that take into account clique complexity, will run additional scalability studies on larger and more complex datasets, and will work with both 3D volumes and 2D images.

Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, through the grant "Scalable Data-Computing Convergence and Scientific Knowledge Discovery," program manager Dr. Laura Biven, and the Center for Applied Mathematics for Energy Research Applications (CAMERA). We also thank the LBNL ALS division.

References

[1] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer Publishing Company, 2013.

[2] L. E. Sucar, *Probabilistic Graphical Models: Principles and Applications*. Springer Publishing Company, 2015.

[3] J. Simmons, C. Przybyla, S. Bricker, D. W. Kim, and M. Corner, "Physics of MRF Regularization for Segmentation of Materials Microstructure Images," in *IEEE International Conference on Image Processing*, 2014, pp. 4882–4886.

[4] X. Yang, F. De Carlo, C. Phatak, and D. Gürsoy, "A convolutional neural network approach to calibrating the rotation axis for X-ray computed tomography," *Journal of Synchrotron Radiation*, vol. 24, no. 2, pp. 469–475, Mar 2017.

[5] C. Hintermüller, F. Marone, A. Isenegger, and M. Stampanoni, "Image processing pipeline for synchrotron-radiation-based tomographic microscopy," *Journal of Synchrotron Radiation*, vol. 17, no. 4, pp. 550–559, Jul 2010.

[6] R.-C. Chen, D. Dreossi, L. Mancini, R. Menk, L. Rigon, T.-Q. Xiao, and R. Longo, "*PITRE*: software for phase-sensitive X-ray image processing and tomography reconstruction," *Journal of Synchrotron Radiation*, vol. 19, no. 5, pp. 836–845, Sep 2012.

[7] M. Khanum, T. Mahboob, W. Imtiaz, H. A. Ghafoor, and R. Sehar, "Article: A survey on unsupervised machine learning algorithms for automation, classification and maintenance," *International Journal of Computer Applications*, vol. 119, no. 13, pp. 34–39, June 2015, full text available.

[8] W. Chen, G. Ostrouchov, D. Pugmire, Prabhat, and M. Wehner, "A Parallel EM Algorithm for Model-Based Clustering Applied to the Exploration of Large Spatio-Temporal Data," *Technometrics*, vol. 55, no. 4, pp. 513–523, 2013.

[9] G. Liu, Z. Lin, X. Tang, and Y. Yu, "Unsupervised object segmentation with a hybrid graph model (hgm)," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 5, pp. 910–924, May 2010.

[10] S. Chen, L. Cso, Y. Wang, J. Liu, and X. Tang, "Image Segmentation by MAP-ML Estimations," in *IEEE Transactions on Image Processing*, 2010, pp. 2254–2264.

[11] A. Jalil, T. Cheema, A. Manzar, and I. Qureshi, "Rotation and gray-scale-invariant texture analysis using radon and differential radon transforms based hidden markov models," *Image Processing, IET*, vol. 4, no. 1, pp. 42–48, February 2010.

[12] A. Huang, R. Abugharbieb, and R. Tam, "A novel rotationally invariant region-based hidden markov model for efficient 3-d image segmentation," in *IEEE Transactions on Image Processing*, vol. 19, no. 10, 2010, pp. 2737–2748.

[13] C. Zheng, Q. Qin, G. Liu, and Y. Hu, "Image segmentation based on multiresolution markov random field with fuzzy constraint in wavelet domain," *Image Processing, IET*, vol. 6, no. 3, pp. 213–221, April 2012.

[14] Q. Zhou, J. Zhu, and W. Liu, "Learning dynamic hybrid markov random field for image labeling," in *IEEE Transactions on Image Processing*, vol. 22, no. 6, 2013, pp. 2219–2232.

[15] O. Lezoray and L. Grady, *Image Processing and Analysis with Graphs: Theory and Practice*. CRC Press, 2012.

[16] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 2, pp. 147–159, Feb 2004.

[17] H. Eslami, T. Kasampalis, and M. Kotsifakou, "A gpu implementation of tiled belief propagation on markov random fields," in *2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013)*, Oct 2013, pp. 143–146.

[18] A. Shekbovstov and V. Hlavac, "A distributed min-cut/maxflow algorithm combining augmentation and push-relabel," in *International Journal of Computer Visualization*, 2012.

[19] O. Jamriska, D. Sykora, and A. Hornung, "A cache-efficient graph cuts on structured grids," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3673–3680.

[20] A. Delong and Y. Boykov, "A scalable graph-cut algorithm for n-d grids," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[21] C. Wang, N. Komodakis, and N. Paragios, "Markov random field modeling, inference, learning in computer vision and image understanding: A survey," *Computer Vision and Image Understanding*, vol. 117, no. 11, pp. 1610 – 1627, 2013.

[22] Z. Meng, D. Wei, A. Wiesel, and A. O. Hero, "Distributed learning of gaussian graphical models via marginal likelihoods," in *The Sixteenth International Conference on Artificial Intelligence and Statistics*, 2013, pp. 39–47.

[23] ——, "Marginal likelihoods for distributed parameter estimation of gaussian graphical models," in *IEEE Transactions on Signal Processing*, vol. 62, no. 20, 2014, pp. 5425–5438.

[24] Y. D. Mizrahi, M. Denil, and N. de Freitas, "Linear and parallel learning of markov random fields," in *Proceedings of International Conference on Machine Learning*, vol. 32, 2014, pp. 1–10.

[25] T. Perciano, D. Ushizima, E. W. Bethel, Y. D. Mizhahi, and J. A. Sethian, "Reduced-complexity Image Segmentation under Parallel Markov Random Field Formulation using Graph Partitioning," in *2016 IEEE International Conference on Image Processing*, Phoenix, AZ, USA, Sep. 2016, lBNL-1005703.

[26] D. Mahapatra and Y. Sun, "Integrating segmentation information for improved mrf-based elastic image registration," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 170–183, Jan 2012.

[27] "Microct." [Online]. Available: microct.lbl.gov

[28] J. Donatelli *et al.*, "Camera: The center for advanced mathematics for energy research applications," *Synchrotron Radiation News*, vol. 28, no. 2, pp. 4–9, 2015.

[29] "Edison." [Online]. Available: http://www.nersc.gov/users/computational-systems/edison/configuration/

[30] K. Moreland and R. Oldfield, "Formal metrics for large-scale parallel performance," in *Proceedings of International Supercomputing Conference*, 2015.