Parallel Computing 53 (2016) 23–31

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Short communication

Parallel simulated annealing using an adaptive resampling interval

Zhihao Lou[a,*], John Reinitz[b]

[a] Department of Computer Science, the University of Chicago, Chicago, Illinois, USA
[b] Department of Statistics, Department of Ecology and Evolution, Department of Molecular Genetics and Cell Biology, Institute of Genomics and Systems Biology, the University of Chicago, Chicago, Illinois, USA

ARTICLE INFO

Article history:
Received 22 June 2015
Revised 11 November 2015
Accepted 2 February 2016
Available online 10 February 2016

Keywords:
Parallel algorithm
Global optimization
Rastrigin function
Evolutionary computation
stochastic optimization
MPI

ABSTRACT

This paper presents a parallel simulated annealing algorithm that is able to achieve 90% parallel efficiency in iteration on up to 192 processors and up to 40% parallel efficiency in time when applied to a 5000-dimension Rastrigin function. Our algorithm breaks scalability barriers in the method of Chu et al. (1999) by abandoning adaptive cooling based on variance. The resulting gains in parallel efficiency are much larger than the loss of serial efficiency from lack of adaptive cooling. Our algorithm resamples the states across processors periodically. The resampling interval is tuned according to the success rate for each specific number of processors. We further present an adaptive method to determine the resampling interval based on the adoption rate. This adaptive method is able to achieve nearly identical parallel efficiency but higher success rates compared to the fixed interval one using the best interval found.

1. Introduction

Simulated annealing, introduced by Kirkpatrick et al. [1], is one of the most widely used general purpose global optimization algorithms. Certain applications in developmental biology and systems biology rely heavily on this particular method of optimization [2–6]. It mimics the physical process of annealing by treating the cost function as an "energy" E and sampling the value of E according to the Boltzmann distribution at some artificial temperature T using the Metropolis algorithm [7]. At each iteration, the algorithm proposes a perturbed state (a "move") with energy E_{new} from the current state with energy E_{old}. If $E_{\text{new}} < E_{\text{old}}$, the new state is accepted. Otherwise the new state is accepted with probability $\exp[-(E_{\text{new}} - E_{\text{old}})/T]$. During the annealing process, T is decreased slowly and uphill moves become less and less likely. If the move generating scheme and schedule for decreasing T are chosen correctly, then as T approaches zero, E will converge to its global minimum. Convergence to the global minimum has been proved for a schedule in which the temperature at the kth iteration $T_k \propto 1/\ln(k)$ and moves are drawn from a Gaussian distribution [8,9], and also for a schedule where $T_k \propto 1/k$ and moves are drawn from a Cauchy distribution [10]. The structure of the proofs reveal that the correct choice of both distribution of the moves and the cooling schedule are important for good performance. In practice, a much faster cooling schedule without a convergence proof was used in both the original Kirkpatrick paper, and most applications. In this schedule, $T_k \propto e^{-\lambda k}$ where λ is sometimes adjusted adaptively based on sampling statistics [11]. Adaptive control of move generation together with

* Corresponding author. Tel.: +1 7738345823.
E-mail addresses: zhlou@uchicago.edu (Z. Lou), reinitz@galton.uchicago.edu (J. Reinitz).

http://dx.doi.org/10.1016/j.parco.2016.02.001
0167-8191/

cooling rate was introduced by Lam and Delosme [12,13]. This innovation turned out to be essential for applications in the natural sciences because the parameters of such problems typically have widely differing characteristic scales [2,14–16].

As computing architectures shift toward more cores and massively parallel computers become more accessible, there have been many attempts to parallelize the simulated annealing algorithm [17]. Because the Metropolis algorithm is a Markov process, early parallelization attempts focused on preserving the single Markov chain in the parallel algorithm by using variants of branch prediction [18–20]. By the nature of these algorithms, they do not scale well as the number of processors increases. The majority of parallelization strategies allow the algorithm to follow multiple Markov chains. These include the division algorithm [21] and variants [22–25], as well as resampling of states [26,27]. There are also algorithms which change the acceptance criterion in the Metropolis algorithm to better suit the parallelization [28,29]. Since Rudolph [30] pointed out the equivalence of simulated annealing and the evolutionary algorithm with a population size of one, explicit hybridization with population based algorithms has been proposed, either by running these algorithms side by side [24,31,32], or by blending in the concepts of evolutionary algorithms with the move generation strategy [30,33–35]. Unfortunately, most published works did not report speedup at all [28–31,34,35], or only reported speedup for up to 16 processors, a small number by today's standards [18,22,24,25,27]. Among those that reported speedup for 32 processors or more, some show mediocre speedup [19,20] while others give a suspiciously super-linear speedup [31,33]. Super-linear speedup suggests either inefficiency in the serial performance or a decrease in the quality of parallel results [36].

The most promising recent development in parallel simulated annealing is reported in recent work on GPUs [37]. In GPUs, it is possible to run an simulated annealing problem on a high number of threads that may exceed the number of cores. The calculation of speedup is made retrospectively by rerunning the problem on a single thread under identical cooling conditions. This leaves the efficiency of the serial annealing process uncontrolled, making comparison with extant methods difficult. We take a different approach in this work, which may profitably combined with architecture-specific methods such as this one in the future.

Among these methods, that proposed by Chu et al. [26] demonstrated practical utility in solving certain estimation problems that arise in systems biology and operations research on the nuclear fuel cycle [3–6,38–46]. This method is based on the serial Lam–Delosme algorithm [12,13], which features an adaptive cooling schedule in which the rate of cooling is a function of the variance and the proportion of accepted moves, together with feedback move generation control. The inner loop of the serial algorithm executes the Metropolis algorithm over τ steps, after which the variance is re-estimated. The mechanism of speedup is to let each processor cool P times faster so that it samples the same number of states in parallel as the serial algorithm would in τ steps over the same temperature interval. To retain the estimation structure of the Lam-Delosme algorithm by parallizing only over the τ loop, the Chu algorithm faces the limitation that it can only work on P processors, where P is an integer divisor of τ, and has a hard limit on scalability at τ processors. In practice the algorithm works the best at $\tau = 100$ and delivers good speedup for up to 50 processors. This method, while apparently the leader among parallel simulated annealing algorithms, thus faces an inherent scalability barrier.

In this work, we examine each component of the algorithm to explore the possibility of scaling beyond 100 processors while delivering good results. We will show that the control of the move distribution is essential in breaking this barrier while the adaptive cooing components of the Chu algorithm are dispensable. We demonstrate our findings using the Rastrigin function as a test problem. The Rastrigin function provides a useful testbed for these investigations because it is difficult to optimize but does not have the heavy CPU requirements found in real world applications.

The rest of this paper is organized as follows. We start with details about the test problem, parameter set, and performance metrics used in this study. We introduce a key communication event in which states can multiply or be annihilated by resampling, and survey the effects that changing the frequency of this resampling step has on the parallel simulated annealing algorithm. Based on these findings, we propose an algorithm to determine the interval of resampling adaptively, and analyze the performance of the proposed algorithm using the metrics we introduced. Finally, we conclude the paper with a discussion about implications and future work.

2. Methods

In this section we discuss the test function, details of the implementation, and how the performance is measured.

2.1. The Rastrigin function

The n-dimensional Rastrigin function [47]

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \tag{1}$$

is a family of multi-modal analytical test functions which is used in many case studies of simulated annealing algorithms [30–32,35]. This family of functions has a global minimum at $f(0,\ldots,0) = 0$ and the number of local minima grows exponentially with n. Since actual scientific applications [3,6] take tens of millions of iterations to find a good result, we choose $n = 5000$ so that similar number of iterations will be required to achieve a good result.