DEPLOYMENT SCENARIOS OF PARALLELIZED CODE IN STOCHASTIC OPTIMIZATION

Günter Rudolph
Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Algorithm Engineering
44221 Dortmund / Germany
Guenter.Rudolph@uni-dortmund.de

Abstract The benefit of using parallel hardware in real-time environments is obvious: For example, if it is necessary to solve some optimization task periodically in a narrow time window a parallelized optimization algorithm can possibly meet the time constraints. In case of deterministic algorithms the situation is clear. But if we use randomized algorithms some questions appear: As randomized algorithms must be run more than once to get a reliable solution we can execute the sequential code in parallel independently or we can execute the parallelized code simultaneously on the parallel hardware in a successive manner. Which approach is better? We analyze several scenarios analytically and offer conditions for deciding when to deploy the parallelized code and when not.

Keywords: Parallel optimization, stochastic optimization, randomized algorithms

1. Introduction

The utility of a parallelized deterministic optimization algorithm is evident: Since the deterministic algorithm is run only once, the parallel version delivers the solution more rapidly. In case of randomized optimization algorithms the situation changes. Typically, these randomized algorithms (RAs) must be run several times to avoid bad results produced by some unlucky sequence of random variables used in the RA. This observation raises the question if the burden of developing a parallel randomized algorithm is worth the effort: Instead of running a parallelized RA several times in sequence on the parallel hardware, one can

also run the original sequential code independently in parallel on several processors. Which are the situations in which running the parallelized code is advantageous? And when the recommendation should be the other way round?

Here, we analyze some situations based on certain scenarios. Our main assumption is that we have a periodically appearing optimization task. Therefore it is reasonable to use the *expectation of random variables* for comparisons: If the expected runtime of successive runs of the parallelized code is less than the expected runtime of parallel runs of the sequential code, then and only then it is advisable to deploy the parallelized RA.

This approach also has the appealing aspect that we can elude from the ongoing discussion how to measure the performance of parallelized RAs [1, 2] in terms of speedup, efficiency and related measures.

Here we extend and generalize our findings presented in [4]. For this purpose, section 2 presents some mathematical results used in the sequel. Sections 3 & 4 present several scenarios and offer conditions for deciding when to deploy the parallelized code and when not. Finally, our conclusions can be found in section 5.

2. Mathematical Preliminaries

Let $X_1, X_2, \ldots, X_p$ be independent and identically distributed (i.i.d.) random variables. Their minimum and maximum are denoted by $X_{1:p} = \min\{X_1, X_2, \ldots, X_p\}$ and $X_{p:p} = \max\{X_1, X_2, \ldots, X_p\}$, respectively. For certain distributions of the X_k the expectation of the minimum and maximum can be calculated analytically. For example [3, p. 35], if the X_k are uniformly distributed in the interval $[\,a, b\,]$ then

$$\mathsf{E}[\,X_k\,] = \frac{b-a}{2}, \quad \mathsf{V}[\,X_k\,] = \frac{(b-a)^2}{12}\,,$$

$$\mathsf{E}[\,X_{1:p}\,] = a + (b-a)\,\frac{1}{p+1} \quad \text{and} \quad \mathsf{E}[\,X_{p:p}\,] = a + (b-a)\,\frac{p}{p+1}\,. \quad (1)$$

Moreover, there exist numerous inequalities for the expectations, each of them based on some assumptions. The most general inequality is probably given in [3, p. 59 & 63] since it only assumes the existence of the second moment.

Theorem 1
Let $X, X_1, X_2, \ldots, X_p$ be i.i.d. random variables with $\mathsf{E}[\,X^2\,] < \infty$. Then

$$\mathsf{E}[\,X\,] - \frac{p-1}{\sqrt{2\,p-1}}\,\mathsf{D}[\,X\,] \le \mathsf{E}[\,X_{1:p}\,] \le \mathsf{E}[\,X_{p:p}\,] \le \mathsf{E}[\,X\,] + \frac{p-1}{\sqrt{2\,p-1}}\,\mathsf{D}[\,X\,]$$

where $\mathsf{D}[\,X\,]$ denotes the standard deviation of X. □

Another result that will be useful is known as Wald's equation. A proof can be found e.g. in [5, p. 166f].

Theorem 2
Let N be a positive, integer-valued random variable and $X_1, X_2, \ldots$ be an i.i.d. sequence of random variables where N is also independent of the X_k. Then the expectation and variance of the random sum consisting of the first N members of the X_k are given by

$$\mathsf{E}\left[\sum_{k=1}^{N} X_k\right] = \mathsf{E}[\,N\,]\cdot\mathsf{E}[\,X_1\,] \tag{2}$$

$$\mathsf{V}\left[\sum_{k=1}^{N} X_k\right] = \mathsf{E}[\,N\,]\cdot\mathsf{V}[\,X_1\,] + \mathsf{V}[\,N\,]\cdot\mathsf{E}[\,X_1\,]^2 \tag{3}$$

where $\mathsf{V}[\,\cdot\,]$ denotes the variance. □

3. Scenario: Run RA Multiple Times, Choose Best Solution Found

In practice, nobody runs a randomized algorithm only once. Rather, the RA is run multiple times and the best solution found within some time limit is used. Figure 1 illustrates our two options how to use the parallel hardware.

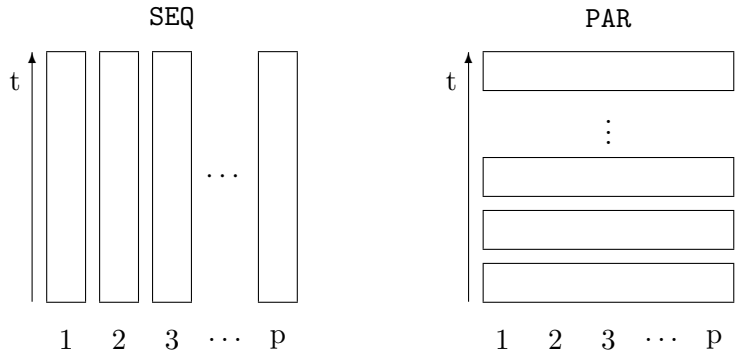

Figure 1. Left: The sequential code is run independently in parallel on p processors. Right: The parallelized code is run on p processors simultaneously for p successive runs.

3.1 Fixed Iteration Number

Let t be the running time of the sequential algorithm and $t_p = c\,t/p$ the running time of the parallelized sequential algorithm, where $c > 1$ aggregates the communication and other overhead costs of the parallelized version. Let n be the maximum number of times we can run the RA before we must use the best solution found and assume that $n = p$ where p is the number of processors.

Then $r = t$ is the total running time of running the sequential algorithm on p processors in parallel. Since the total running time of p sucessive runs of the parallelized version is $r_p = p \times t_p = c\,t$ we can see easily that nothing is gained by a parallelization. Even worse, every effort invested in this task is a waste of resources.

3.2 Random Iteration Number

The situation changes if the running time of the RA is a random variable. For instance, this may be caused by some stopping rule that is independent from the iteration counter. Let T be the random running time of the sequential algorithm and $T_p = c\,T/p$ the running time of the parallelized sequential algorithm with $c > 1$. Again, assume $n = p$. Then the random total running time R of running the sequential algorithm on p processors in parallel is

$$R = \max\{\, T(1), T(2), \ldots, T(p) \,\} = T_{p:p}$$

where $T(i)$ is the running time at processor i. Clearly, the $T(i)$ are independent and identically distributed. Assume that $T(i)$ is normally distributed with mean $t > 0$ and variance σ^2. Then the expectation of R can be approximated [3] via

$$\mathsf{E}[\, R \,] = \mathsf{E}[\, T_{p:p} \,] \approx \mathsf{E}[\, T \,] + \mathsf{D}[\, T \,] \sqrt{2 \log p}\,. \tag{4}$$

The random total running time R_p of p sucessive runs of the parallelized version is given by

$$R_p = \sum_{i=1}^{p} T_p(i) = \frac{c}{p} \sum_{i=1}^{p} T(i)$$

with expectation

$$\mathsf{E}[\, R_p \,] = c\,\mathsf{E}[\, T \,]\,.$$

Thus, the parallelized version is faster if

$$\mathsf{E}[\, R_p \,] < \mathsf{E}[\, R \,] \Leftrightarrow c < 1 + \frac{\mathsf{D}[\, T \,]}{\mathsf{E}[\, T \,]} \times \sqrt{2 \log p}\,. \tag{5}$$

In other words, the larger is the coefficient of variation $\nu = \mathsf{D}[\,T\,]/\mathsf{E}[\,T\,]$ the larger the benefit achieved by the parallelization of the sequential algorithm! As seen from this analysis, this scenario can be an appropriate field of deployment of parallelized RAs.

One may object that the conclusions drawn from the relationship in (5) are shaky since equation (4) is an approximation only. In order to invalidate this objection we first consider an example for which the result can be reproduced exactly in analytical manner. Next we generalize the result by means of Theorem 1.

Assume that $T(i) \sim U(t-\varepsilon, t+\varepsilon)$ are uniformly distributed in the interval $[\,t-\varepsilon, t+\varepsilon\,]$ for some $t, \varepsilon > 0$. For sake of brevity we shall write T instead of $T(i)$. Insertion in (1) yields

$$\mathsf{E}[\,T\,] = t\,, \quad \mathsf{V}[\,T\,] = \frac{\varepsilon^2}{3}\,, \quad \mathsf{E}[\,T_{p:p}\,] = t + \varepsilon\,\frac{p-1}{p+1}\,.$$

Thus, $\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,]$ if and only if $c\,t \le t + \varepsilon\,(p-1)/(p+1)$ or equivalently

$$c < 1 + \frac{\varepsilon}{t\,\sqrt{3}}\,\frac{p-1}{p+1}\,\sqrt{3} \;=\; 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \;\times\; \frac{p-1}{p+1}\,\sqrt{3}\,. \tag{6}$$

For example, if we use 9 processors and the running time is uniformly distributed between 40 and 60 seconds then (6) yields $c < 1 + 4/25 = 1.16$. As a consequence, the efficiency $1/c$ of the parallelization must be larger than $25/29 \approx 86.2$ %. Otherwise, one should run the sequential code in parallel independently.

Next, we generalize our findings. Comparison of (5) and (6) reveals the same pattern:

$$c < 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \;\times\; g(p) \tag{7}$$

for some function $g(\cdot)$ depending on the number of processors p. In order to derive condition (7) analytically recall that the condition originally reads

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \quad \Leftrightarrow \quad c\,\mathsf{E}[\,T\,] < \mathsf{E}[\,T_{p:p}\,] \quad \Leftrightarrow \quad c < \frac{\mathsf{E}[\,T_{p:p}\,]}{\mathsf{E}[\,T\,]}\,.$$

Evidently, this condition is fulfilled if we bound $\mathsf{E}[\,T_{p:p}\,]$ from above via Theorem 1, that is valid for arbitrary runtime distributions. We obtain

$$c < \frac{\mathsf{E}[\,T_{p:p}\,]}{\mathsf{E}[\,T\,]} \le \frac{\mathsf{E}[\,T\,] + \mathsf{D}[\,T\,] \times \frac{p-1}{\sqrt{2\,p-1}}}{\mathsf{E}[\,T\,]} = 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \;\times\; \frac{p-1}{\sqrt{2\,p-1}}$$

confirming that the pattern in (7) did not appear by chance. Moreover, we have shown that

$$g(p) \le \frac{p-1}{\sqrt{2\,p-1}}$$

regardless of the runtime distribution of T.

4. Scenario: Run Until Satisfactory Solution Found

One might argue that the previous scenario is not always the case. For example, if we need only a satisfactory solution then we can stop the RA as soon as such a solution has been detected. In principle, this can happen in a single run of the RA. Figure 2 illustrates our two options how to use the parallel hardware.

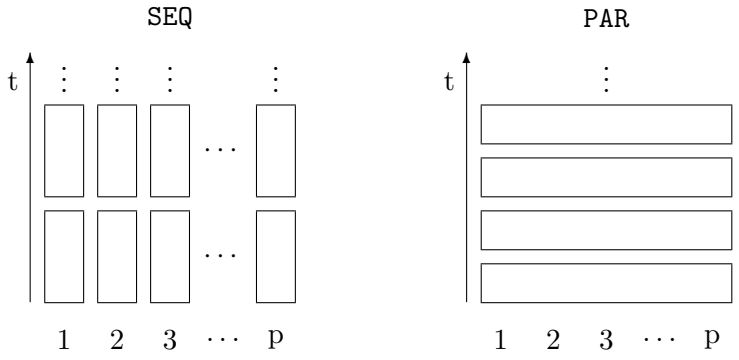

Figure 2. Left: The sequential code is run independently in parallel on p processors until a satisfactory solution is found. Right: The parallelized code is run repeatedly on p processors simultaneously until a satisfactory solution is found.

4.1 Fixed Iteration Number

As in the previous scenario let t be the running time of the sequential algorithm and $t_p = c\,t/p$ the running time of the parallelized sequential algorithm with $c > 1$. Suppose there exists a success probability $s \in (0, 1)$ for each run of the RA such that the random variable G represents the number of runs until a successful run occurs. The random variable G has geometrical distribution with probability function

$$\mathsf{P}\{\, G = k \,\} = s\,(1-s)^{k-1}$$

for $k = 1, 2, \ldots$ and $s \in (0, 1)$ with

$$\mathsf{E}[\, G \,] = \frac{1}{s} \quad \text{and} \quad \mathsf{V}[\, G \,] = \frac{1-s}{s^2}\,.$$

The time until a successful run occurs on a single processor is $S = t\,G$. Therefore, the random total running time R of running the sequential

algorithm on p processors in parallel is

$$R = \min\{\, S(1), S(2), \ldots, S(p) \,\} = S_{1:p} = t\, G_{1:p}$$

where $G_{1:p}$ denotes the minimum of p independent and identically distributed geometrical random variables. According to [6] we have

$$\mathsf{E}[\, G_{1:p} \,] = \frac{1}{1-(1-s)^p} \quad \text{and} \quad \mathsf{V}[\, G_{1:n} \,] = \frac{(1-s)^n}{[\, 1-(1-s)^n \,]^2}$$

such that

$$\mathsf{E}[\, R \,] = t\, \mathsf{E}[\, G_{1:p} \,] = \frac{t}{1-(1-s)^p}\,.$$

The random total running time R_p of p sucessive runs of the parallelized version is given by

$$R_p = t_p\, S = \frac{c}{p}\, t\, S$$

with expectation

$$\mathsf{E}[\, R_p \,] = \frac{c}{p}\, t\, \mathsf{E}[\, S \,] = \frac{c\, t}{s\, p}\,.$$

Since

$$\mathsf{E}[\, R_p \,] < \mathsf{E}[\, R \,] \Longleftrightarrow c < \frac{s\, p}{1-(1-s)^p}$$

there are constellations in which a parallelized version is useful. Figure 3 is intended to provide an impression about the interrelationships. For small success probabilities s as one usually faces in optimizations task in which RAs are used as last remedy, the effeciency of the parallel implementation must be extremely high for recommending the deployment of the parallelized code. Especially in real-time environments assumed here it is unlikely to achieve such a high efficiency.

4.2 Random Iteration Number

Let $T(i)$ be the random running time of run i. Then

$$S = \sum_{i=1}^{G} T(i)$$

is the random time until the first successful run on a single processor. According to Theorem 2 we have $\mathsf{E}[\, S \,] = \mathsf{E}[\, G \,]\, \mathsf{E}[\, T \,]$. As a consequence, the random total running time R of running the sequential algorithm on p processors in parallel is

$$R = \min\{\, S(1), S(2), \ldots, S(p) \,\} = S_{1:p}$$

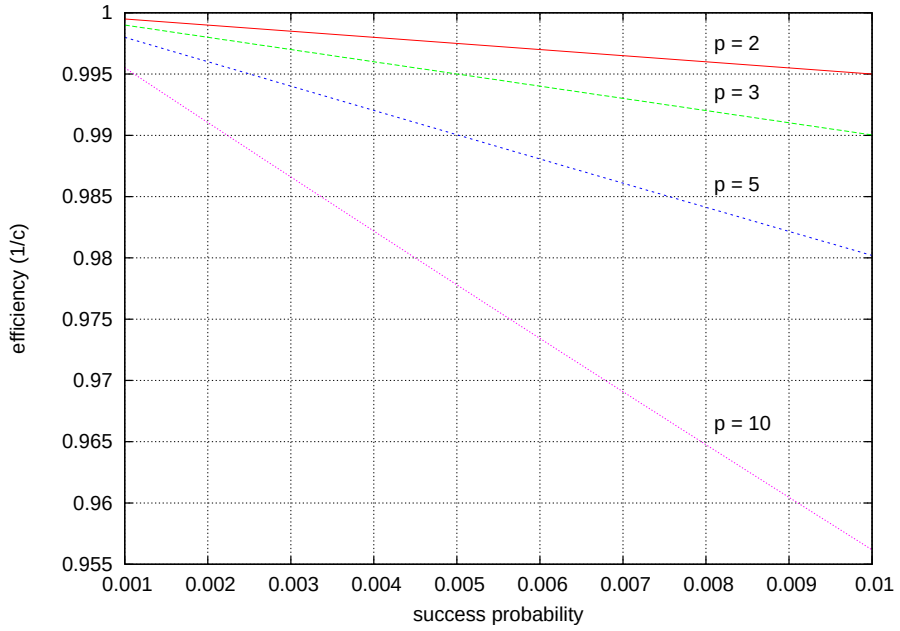

Figure 3. Success probability s versus efficiency $1/c$ of the parallel implementation for some processor numbers.

with

$$\mathsf{E}[\,R\,] = \mathsf{E}[\,S_{1:p}\,] < \mathsf{E}[\,S\,] = \mathsf{E}[\,T\,]\,\mathsf{E}[\,G\,]\,.$$

The random total running time R_p of p sucessive runs of the parallelized version is given by

$$R_p = \sum_{i=1}^{G} T_p(i) = \frac{c}{p} \sum_{i=1}^{G} T(i)$$

with

$$\mathsf{E}[\,R_p\,] = \frac{c}{p}\,\mathsf{E}[\,T\,]\,\mathsf{E}[\,G\,] = \frac{c}{p}\,\mathsf{E}[\,S\,] = \frac{c\,t}{s\,p}\,.$$

Our condition reads

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \quad \Leftrightarrow \quad \frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S_{1:p}\,]\,.$$

We can express $\mathsf{E}[\,S\,]$ in terms of $\mathsf{E}[\,T\,]$ and $\mathsf{E}[\,G\,]$ but there is a problem for $\mathsf{E}[\,S_{1:p}\,]$: Although we can use the lower bound of Theorem 1 to claim that there is a nonnegative-valued function $h(\cdot)$ with $\mathsf{E}[\,S_{1:p}\,] =$

$\mathsf{E}[\,S\,] - \mathsf{D}[\,S\,] \times h(p)$ and we can express $\mathsf{D}[\,S\,]$ in terms of moments of T and G via Theorem 2, the resulting formula

$$\frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S\,] - \mathsf{D}[\,S\,] \times h(p) \quad \Leftrightarrow \quad c < p\left(1 - \frac{\mathsf{D}[\,S\,]}{\mathsf{E}[\,S\,]} \times h(p)\right)$$

does not yield much insight for analyzing the situation.

Therefore we take a look at our condition $\frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S_{1:p}\,]$ again. If each T_i has a minimum runtime $a > 0$ then $\mathsf{E}[\,S\,] \geq a\,\mathsf{E}[\,G\,]$ and $\mathsf{E}[\,S_{1:p}\,] \geq a\,\mathsf{E}[\,G\,]$. Since

$$\frac{c}{p}\,\mathsf{E}[\,S\,] \geq \frac{c}{p}\,a\,\mathsf{E}[\,G\,] \to 0 \text{ as } p \to \infty$$

whereas

$$\mathsf{E}[\,S_{1:p}\,] \geq a\,\mathsf{E}[\,G\,] > 0 \text{ regardless of } p$$

we may conclude that there exists a processor number p_0 such that $\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,]$ for all $p > p_0$. Thus, this scenario is well suited for parallelized code if many processors are available.

5. Conclusions

We have shown that the recommendation for a deployment of parallelized code depends on several constraints. If we have a fixed time slot and a constant running time of the algorithm then the deployment of parallelized code is a waste of resources. If we can wait until completion of the randomized algorithm which has a random running time, then we need a moderately efficient parallel implementation and a large variation in the running time to favor the parallelized code. If we are in the situation to repeat the algorithm until it fulfills some criterion, then the condition for deploying parallelized code demands a hardly achievable efficiency of the code in case of *constant* running time. If the running time is random then parallelized code may lead to shorter overall running time if many processors are available. The theory in its current state, however, does not yet provide a condition to quantify the number of processors that must be available. Nevertheless, the theory provides some clues that random running times of the randomized algorithms more often lead to recommendations for deploying parallelized code.

References

[1] J. Aczél and W. Ertel. A new formula for speedup and its characterization. *Acta Informatica*, 34:637–652, 1997.

[2] E. Alba and A. Luque. Measuring the performance of parallel metaheuristics. In E. Alba, editor, *Parallel metaheuristics: A New Class of Algorithms*, pages 43–62, Hoboken (NJ), 2005. Wiley.

[3] H. A. David. *Order Statistics*. Wiley, New York, 2nd edition, 1981.

[4] G. Rudolph. Parallel evolution strategies. In E. Alba, editor, *Parallel metaheuristics: A New Class of Algorithms*, pages 155–169, Hoboken (NJ), 2005. Wiley.

[5] K. D. Schmidt. *Versicherungsmathematik*. Springer, Berlin et al., 2002.

[6] D. H. Young. The order statistics of the negative binomial distribution. *Biometrika*, 57(1):181–186, 1970.