Combinatorial Search from an Energy Perspective

Mohamed Siala[a,*], Barry O'Sullivan[b,*]

[a] *LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France*
[b] *Insight Centre for Data Analytics and Confirm Centre for Smart Manufacturing, School of Computer Science & IT, University College Cork, Ireland*

Abstract

Most studies related to parallel and portfolio search for solving combinatorial problems, such as those found in Boolean satisfiability or constraint programming, evaluate search cost in terms of runtime. However, given the complex computing architectures available today and the focus on the environmental impact of computing, there is growing interest in also considering the energy cost associated with solving these problems. In the context of combinatorial problem-solving, a simple approximation of energy cost is the product of the number of machines multiplied by the runtime spent to solve a problem instance. However, the picture is much more complex due to the impact that the distribution of runtimes, even for solving a single specific instance, can have on search cost. In this paper we present an initial, but comprehensive, study on the impact of runtime distribution on the amount of energy required for combinatorial problem solving characterized by two common continuous runtime distributions, namely the Weibull and Pareto distributions. The primary contribution of this paper is to demonstrate that there is an interesting and non-trivial relationship between runtime, parallelisation, and energy cost in combinatorial solving that is worthy of further study.

Keywords: Randomized algorithms, parallel algorithms, combinatorial optimisation, probability distributions.

*Corresponding author

Email addresses: `mohamed.siala@laas.fr` (Mohamed Siala), `b.osullivan@cs.ucc.ie` (Barry O'Sullivan)

1. Introduction

Cloud computing is ubiquitous in solving combinatorial problems, whereby a number of machines are rented, or shared from a common pool, to solve a particular instance using parallel or portfolio search. Clearly, the more machines used, the faster a solution might be found. However, this will increase the cost of renting the cloud system, as well as the amount of energy used in the process. The search cost in this context is typically proportional to the *energy* consumed where the energy can be approximated by the number of cores (i.e., machines) used multiplied by the runtime [1, 2]. While many studies in the literature focus on the time speed-up with parallel and portfolio search, little is known in terms of energy consumption.

Modern combinatorial solvers are typically randomized and are known to exhibit a variance in runtime that can be extremely large [3]. Specifically, the runtime that a randomised algorithm A (Las Vegas type [4]) takes to solve an instance I can be very different using two different random seeds. Researchers have used the notion of "runtime distribution" to describe and study this phenomenon [3, 4, 5, 6, 7]. It turns out that the variation in runtime of combinatorial problems can often be characterised by heavy-tailed distributions [3].

Runtime distributions have a long history in combinatorial search [3, 6, 7, 8]. Previous work focuses on the impact of such distributions on the solution time and the time speed-up in parallel seach [5, 7]. Moreover, recently, machine learning techniques were developped for predicting the runtime distributions [1, 2, 8, 9, 10]. In this paper we consider the impact of runtime distributions from an energy perspective. Our objective is to understand how runtime distribution affects the energy required to solve combinatorial problems. We consider two of the most common continuous runtime distributions: Weibull and Pareto. Both have been shown to usefully characterise runtime distributions [3, 11]. A Weibull distribution is denoted by $W(\lambda, b)$ where $\lambda \in (0, +\infty)$ is the scale parameter and $b \in (0, +\infty)$ is the shape parameter. The cumulative distribution function (CDF) of $W(\lambda, b)$ is $F(x) = 1 - e^{-(\frac{x}{\lambda})^b}$ for all $x \in [0, +\infty)$. The Weibull distribution can model exponential, heavy-tailed, and light-tailed distributions: if $b < 1$, the distribution is heavy-tailed; if $b > 1$, the distribution is light-tailed; and, if $b = 1$ the distribution is exponential. A Pareto distribution $P(m, \alpha)$, where $m \in (0, +\infty)$ is called the scale parameter and $\alpha \in (0, +\infty)$ is called the shape parameter, is defined with a cumulative distribution function $F(x) = 1 - (\frac{m}{x})^\alpha$ where $x \geq m$. Figure 1 gives the log-log plot of the survival function (1- CDF) of different Weibull and Pareto distributions.

Our analysis of time and energy variation shows that while the expected time of parallel search is a decreasing function in the number of CPU cores, energy consumption can, theoretically, exhibit a diverse set of behaviours depending on the runtime distribution of the instance. We prove that Pareto distributions always have an optimal number of cores that minimises energy consumption that one can calculate. However, energy cost associated with runtimes characterised by Weibull distributions can either be decreasing (heavy-tailed case), constant (exponential case), or increasing (light-tailed case). In addition to energy variation, we compute the exact time speed-up for both types of distribution. In terms of expected time and time speed-up, we show that parallel Pareto distributions are bounded, whereas Weibull distributions are not.

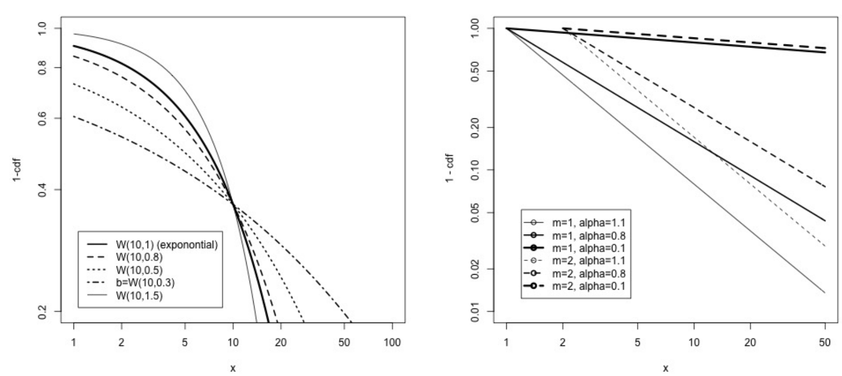

Figure 1: Survival Function (1-CDF) for Different Weibull (left) and Pareto (right) Distributions

2. Expected Time and Energy Consumption

We consider combinatorial search with a Las Vegas algorithm (randomized algorithm that always produces the correct answer when it stops but its running time is a random variable [4]). Let I be a decision problem instance and A be a randomized algorithm to solve I (i.e., to decide I and eventually to find a solution). Let $k \geq 1$ be an integer. We denote by $Parallel(A, k, I)$ the search algorithm that runs algorithm A k independent times to solve I. $Parallel(A, k, I)$ stops when one of the k runs solves I. Suppose that A follows a (continuous) probability distribution X when solving I (in terms of runtime). Denote by $f(x)$ its probability density function (PDF) and by $F(x)$ its cumulative distribution function (CDF). We assume that there exists $x_0 \in [0, +\infty)$ such that $f(x) = F(x) = 0$ for all $x \leq x_0$. We denote by Y_k the probability distribution of the runtime of $Parallel(A, k, I)$. Let $g_k(x)$ be the PDF of Y_k and let $G_k(x)$ be the CDF of Y_k. Let $T(k)$ be the expected time associated with $Parallel(A, k, I)$. Inspired by the study of [1] on finite and discrete distributions, we define the energy consumption of $Parallel(A, k, I)$ to be $E(k) = k \times T(k)$. In the remainder of this paper, we use the energy $E(k)$ as a function of the number of runs k. Proposition 1 gives the exact formula for energy consumption in parallel search.

Proposition 1. *(Energy as a function of the number of cores/runs k)* $E(k) = k^2 \times \int_{x_0}^{\infty} x \times (1 - F(x))^{(k-1)} \times f(x)\ dx$.

Proof. We can adapt the formulae of the expected time from [7] to compute $E(k)$. Since the parallel runs are independent, then Y_k satisfies $P[Y_k > x] = P[X > x]^k$. Hence $1 - G_k(x) = P[X > x]^k$. Therefore $G_k(x) = 1 - (1 - F(x))^k$. We can also infer that $g_k(x) = k \times (1 - F(x))^{(k-1)} \times f(x)$. Since the expected value of Y_k is $\int_{x_0}^{\infty} x \times g_k(x)\ dx$, then $T(k) = k \times \int_{x_0}^{\infty} x \times (1 - F(x))^{(k-1)} \times f(x)\ dx$. Hence the result. □

There are two observations that should be taken into account regarding energy consumption. First, despite the fact that the expected time of $Parallel(A, k, I)$

is always decreasing as a function of the number of cores (k), the energy consumption is not necessarily monotonic. A counter-example is given later with Pareto distributions (see Figure 7). Second, theoretically, since $E(k) = k \times T(k)$ and $T(k)$ is a mean of a distribution, then it might be case that the value of expected energy is infinite or might not exist. We define the notion of *time speed-up* based on [5, 7]. Let k_1 be the first integer such that $T(k)$ is finite. If k_1 exists, then we define the time speed-up for every $k > k_1$ to be $\frac{T(k_1)}{T(k)}$.

3. Runtimes Characterised by Weibull Distributions

Recall that the cumulative distribution function of a Weibull distribution $W(\lambda, b)$ is $F(x) = 1 - e^{-(\frac{x}{\lambda})^b}$ for all $x \in [0, +\infty)$, i.e. $x_0 = 0$. The probability density function of $W(\lambda, b)$ is given by $f(x) = \frac{b}{\lambda} \times (\frac{x}{\lambda})^{(b-1)} \times e^{-(\frac{x}{\lambda})^b}$ for all $x \in [0, +\infty)$. Let Γ be the Gamma function: $\Gamma(z) = \int_0^\infty u^{z-1} \times e^{-u}\, du$. Proposition 2 gives the (expected) energy function for $W(\lambda, b)$.

Proposition 2. *The energy function of $W(\lambda, b)$ is $E(k) = k^{1-\frac{1}{b}} \lambda \Gamma(1 + \frac{1}{b})$.*

Proof. (Sketch) $E(k) = k^2 \frac{b}{\lambda^b} \int_0^\infty x^b \times e^{-\frac{k}{\lambda^b} x^b}\, dx$. Let $u = \frac{k}{\lambda^b} x^b$. Thus, $du = \frac{bk}{\lambda^b} x^{b-1} dx$, and $E(k) = k^2 \times \frac{b}{\lambda^b} \int_0^\infty \frac{\lambda^b}{bk} \frac{\lambda}{k^{\frac{1}{b}}} u^{\frac{1}{b}} \times e^{-u}\, du = k^{1-\frac{1}{b}} \lambda \Gamma(1 + \frac{1}{b})$. □

Proposition 3 describes the energy behavior of a Weibull distribution.

Proposition 3. *The energy function for $W(\lambda, b)$ is: decreasing if $b < 1$, i.e., heavy-tailed; increasing if $b > 1$, i.e., light-tailed; and constant otherwise, i.e., exponential distribution.*

Proof. Obviously if $b = 1$, the energy $E(k) = \lambda \times \Gamma(1 + \frac{1}{b})$ for all $k \geq 1$. If $b \neq 1$, then $E'(k) = (1 - \frac{1}{b}) \times \lambda \times \Gamma(1 + \frac{1}{b}) \times k^{-\frac{1}{b}}$. Therefore, E is either decreasing or increasing. The result is immediate from the fact that $E(2) > E(1)$ when $b > 1$ and $E(2) < E(1)$ when $b < 1$. □

Proposition 4 gives the time speed-up for $W(\lambda, b)$, proven by construction.

Proposition 4. *The time speed-up of $W(\lambda, b)$ is $\frac{T(1)}{T(k)} = k^{\frac{1}{b}}$ for all $k \geq 2$.*

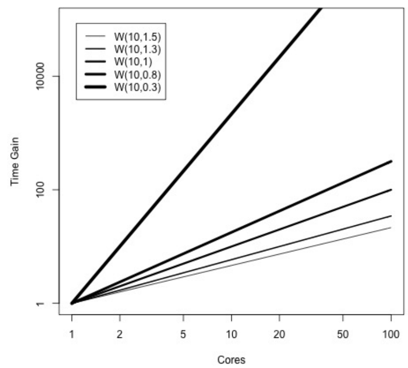

Figure 2: Time Speed-up for $W(\lambda, b)$

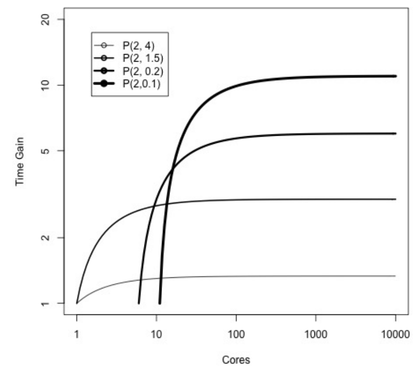

Figure 3: Time Speed-up for $P(m, \alpha)$

Proposition 4 clearly shows that the time speed-up function is not bounded for Weibull distributions. This is not the case for Pareto distributions as we show in the next section. We give the log-log plot of the time speed-up function in Figure 2 for a visual interpretation for different Weibull distributions.

Figure 4 gives the log-log plots of expected runtime as a function of the number of cores for different Weibull distributions, from light-tailed to heavy-tailed distributions. Figure 5 plots the energy consumption for the same runtime distributions. Figure 4 shows that the "heavier" the runtime distribution, i.e., the smaller the value of the shape parameter b, the more beneficial parallel search is in terms of solution time. This is particularly true for time speed-up (Proposition 4). When it comes to energy consumption (Figure 5), we find three different behaviours. First, when $b = 1$ (i.e., exponential distribution), energy consumption is constant, as we showed in Proposition 3. Second, in the case of heavy-tailed runtime distribution ($b < 1$), energy consumption is decreasing. In other words, the heavier the distribution, the better parallelism helps in reducing overall energy consumption. Last, if the runtime distribution is light-tailed ($b > 1$), then parallelism clearly makes energy consumption worse. In fact, the "lighter", i.e., the bigger the shape b, the runtime distribution, the worse energy consumption becomes. This makes intuitive sense since there is far less variation in runtime, and using additional cores has no benefit in terms of overall runtime.

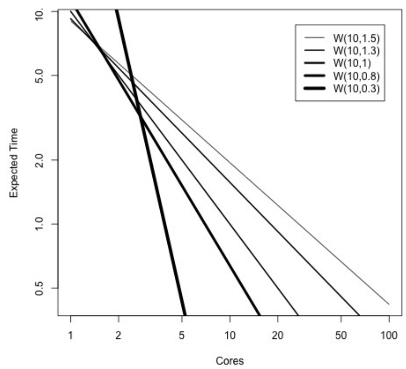

Figure 4: Expected Time for $W(\lambda, b)$

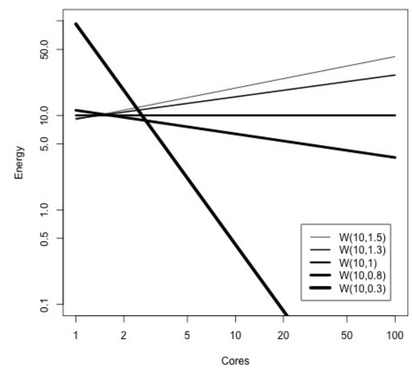

Figure 5: Energy Function for $W(\lambda, b)$

4. Runtimes Characterised by Pareto Distribution

A Pareto distribution $P(m, \alpha)$ is defined with a probability density function $f(x) = \frac{\alpha m^\alpha}{x^{\alpha+1}}$ and a cumulative distribution function $F(x) = 1 - (\frac{m}{x})^\alpha$ where $x \geq m$ (see Figure 1).

Proposition 5. *The energy function of $P(m, \alpha)$ is $E(k) = \frac{k^2 \alpha m}{\alpha k - 1}$ when $\alpha k > 1$ and infinite otherwise.*

Proof. (Sketch) $E(k) = k^2 \alpha m^{\alpha k} \int_m^\infty x^{-\alpha k}\, dx$. If $\alpha k = 1$, then $E(k) = +\infty$. Suppose that $\alpha k \neq 1$, then $E(k) = k^2 \alpha m^{\alpha k} \frac{1}{-\alpha k + 1} [x^{-\alpha k + 1}]_m^\infty$. If $\alpha k < 1$ then $E(k) = +\infty$. When $\alpha k > 1$, we have $E(k) = \frac{k^2 \alpha m}{\alpha k - 1}$. □

Let $k_1 = \min\{k \geq 1 \in \mathbb{N} \mid \alpha k > 1\}$. In the following, $T(k)$ and $E(k)$ are defined for $k \geq k_1$, otherwise they are infinite.

Proposition 6. *The energy function of $P(m, \alpha)$ has a unique minimum: at $k = \frac{2}{\alpha}$ if $\frac{2}{\alpha} > k_1$; and at k_1 otherwise.*

Proof. (Sketch) If $\alpha k - 1 > 0$, then $E'(k) = \frac{\alpha m k * (\alpha k - 2)}{(\alpha k - 1)^2}$. We have $E'(k) = 0$ iff $k = \frac{2}{\alpha}$. Moreover $\forall k < \frac{2}{\alpha}, E'(k) < 0$ and $\forall k > \frac{2}{\alpha}, E'(k) > 0$. Therefore, if $\frac{2}{\alpha} \geq k_1$, $E(k)$ has a unique minimum at $k = \frac{2}{\alpha}$. Otherwise, the minimum is at k_1, i.e., the energy is increasing. □

Proposition 7 gives the time speed-up. The proof is by construction.

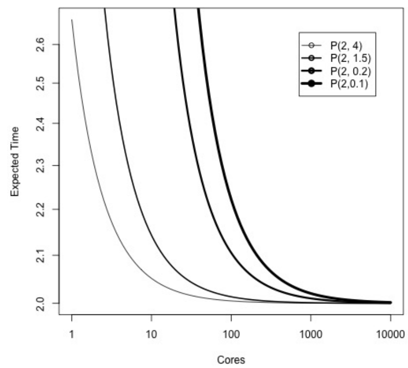

Figure 6: Expected Time for $P(m, \alpha)$

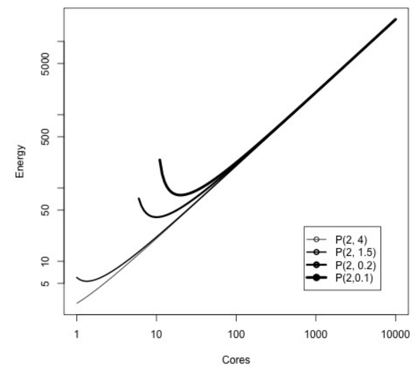

Figure 7: Energy Function for $P(m, \alpha)$

Proposition 7. *The time speed-up function of a parallel Pareto distribution* $P(m, \alpha)$ *is* $\frac{T(k_1)}{T(k)} = \frac{k_1 \times (\alpha k - 1)}{k(\alpha k_1 - 1)}$ *for all* $k > k_1$.

From Proposition 7 one can deduce that the time speed-up cannot exceed $\frac{\alpha k_1}{(\alpha k_1 - 1)}$. Observe also that the expected time is also bounded: $T(k) > m$. We give the log-log plot of the time speed-up function in Figure 3 for different Pareto distributions.

We plot the expected time in Figure 6 for different Pareto distributions. The corresponding energy consumption is plotted in Figure 7. These figures confirm exactly the observations that we made about the upper bound of the expected time and the existence of the minimum point of energy consumption.

5. Conclusion & Discussion

We have studied energy variation in parallel search algorithms via two of the most common continuous runtime distributions: Weibull and Pareto. Our theoretical study showed that while the expected runtime of parallel search is a decreasing function of the number of cores, energy consumption has diverse behaviour depending on the runtime distribution of the problem at hand. On the one hand, we showed that Pareto distributions always have an optimal number of cores for energy consumption that one can calculate. On the other hand, we proved that the energy function with Weibull distributions can either be decreasing (heavy-tailed), constant (exponential), or increasing (light-tailed).

Our study provides a basis for understanding the relationship between time and energy consumption in parallel search. Our results can be used in parallel solving to estimate the energy consumption. Indeed, by predicting the runtime distribution [8], one can use the energy formulas we proposed to predict its variation and its relationship with the solution time.

It would be interesting in the future to consider recent advances in parallel and portfolio solving from an energy perspective. Indeed, modern approaches often use diverse sets of algorithms that might share information. Furthermore, often when restarting search, the search algorithm might use information collected during previous executions. We expect it to be challenging to formally incorporate how such information can impact the energy consumption of search.

Acknowledgements

This publication is supported by Science Foundation Ireland under Grants 12/RC/2289 and 16/RC/3918, which are co-funded under the European Regional Development Fund.

References

[1] B. Hurley, D. Mehta, B. O'Sullivan, Elastic solver: Balancing solution time and energy consumption, CoRR abs/1605.06940. `arXiv:1605.06940`.

[2] B. Hurley, Exploiting machine learning for combinatorial problem solving and optimisation, Ph.D. thesis, University College Cork (2016). URL `https://cora.ucc.ie/handle/10468/5374`

[3] C. P. Gomes, B. Selman, N. Crato, H. A. Kautz, Heavy-tailed phenomena in satisfiability and constraint satisfaction problems, J. Autom. Reasoning 24 (1/2) (2000) 67–100.

[4] M. Luby, A. Sinclair, D. Zuckerman, Optimal speedup of las vegas algorithms, Information Processing Letters 47 (4) (1993) 173 – 180.

[5] T. Hogg, C. P. Williams, Expected gains from parallelizing constraint solving for hard problems, in: Proceedings of AAAI 94, 1994, pp. 331–336.

[6] R. M. Aiex, M. G. C. Resende, C. C. Ribeiro, Probability distribution of solution time in GRASP: an experimental investigation, J. Heuristics 8 (3) (2002) 343–373.

[7] C. Truchet, A. Arbelaez, F. Richoux, P. Codognet, Estimating parallel runtimes for randomized algorithms in constraint solving, J. Heuristics 22 (4) (2016) 613–648.

[8] A. Arbelaez, C. Truchet, P. Codognet, Using sequential runtime distributions for the parallel speedup prediction of SAT local search, TPLP 13 (4-5) (2013) 625–639.

[9] F. Hutter, L. Xu, H. H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: Methods & evaluation, Artif. Intell. 206 (2014) 79–111.

[10] K. Eggensperger, M. Lindauer, F. Hutter, Neural networks for predicting algorithm runtime distributions, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden., 2018, pp. 1442–1448.

[11] D. Frost, I. Rish, L. Vila, Summarizing CSP hardness with continuous probability distributions, in: Proceedings of AAAI'97/IAAI'97, AAAI Press, 1997, pp. 327–333.