Energy efficiency for computing systems

Keywords: Computational Complexity Theory; Information Theory; Thermodynamic Computing

Matheus Sant Ana Lima

Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing

santanalima.matheus@gmail.com

AFFILIATION Department of Computer Science, Federal University of São Carlos, Sao Paulo, Brazil

Abstract

Functions per second produced by a program is relevant to halting theory and the computational complexity of a decision problem. Strings and state function transactions are an important measurement of computer performance.

The conventional computational complexity models describe the resources limitations in terms of time and memory required to process a random string and other data structures. In algorithmic information theory, however, the string complexity is described in terms of the limitations related to the language used to program an arbitrary decision problem and the statistical distribution of input and output strings. In this paper we look for a more fundamental limit: the thermo-computational constraints related to process information.

A Turing Machine instantiation such as a Carnot Engine processing signals is transforming information within a discrete, thermodynamic physical system that has storage regions of high and low entropy. The energy source is converted to string computing effort but is also lost as heat and noise, such observed with thermal and quantum noise.

The energy efficiency of a computation-communication system can be measured in Joule per bits (J/bits) and Bits per second (Bits/s). The proposed general optimization challenge is how can any arbitrary binary program be improved in terms of the energy change rate and final entropy state, computed by a machine with finite energy constrains.

In other words, this can be summarized as *"what is the minimal volume of energy change required to describe and compute a bit of information".*

In this work we have applied this concept to optimize the search for the optimal network routing path in a set of distributed and interconnected machines nodes with an arbitrary topology. The results obtained from the experiment provides a better string optimization factor with an average information gain performance of 11.52% versus 8.53% from benchmark. The expected energy difference between the proposed QA and the benchmark SA algorithms is approximately 8.11%. The findings and proposed framework can be expanded to the energy efficiency of other non-polynomial problems.

Introduction

The proposed framework allows the statistical analyses of the energy efficiency and string complexity of heuristics approximation algorithms. Those methods are used by program scheduling and routing optimization in transactions across a network of machines.

The research reveals for a small number of inputs strings the traditional nature-inspired heuristics simulations such as Simulated Annealing, Neural Networks and Genetic Algorithms have an acceptable performance but for large scale search space, the energy limits become important for many engineering applications such as Distributed Computing (DC) and in those cases a statistical modeling is required to achieve greater entropy optimization.

Within the context of DC architectures like for example in remote interconnected datacenters for instance in Cloud Computing or local, self-managed and owned, infrastructure hosting, there are three units of cost measurement described in Table 1

DC Type	Description	Utility function – Associated Cost
Private owned data center, self-managed.	It's also named as on-premises facilities or edge computing. The cost to construct, deploy and maintain equipment's and machinery related to manage a	Measured as dollars (USD) per M^2

	data center and own its physical premises.	
Remote off-site, Cloud Computing	The subscription cost estimated to allocate some processing, storage, and network capacity for a period in a shared group of remote external resources.	Measured in dollar (USD) per second
Collocation Facility	The rent cost to use some external facility provider with an energy source installed. This is used to build a private data center unit.	Measured as dollar per kVA

Table 1 Types of Distributed Computing models

In this paper we have developed a framework for energy efficiency analysis applied to process scheduling and network routing for a distributed group of machines over a network of independent nodes.

In a distributed computing model, the machines are connected over a network of data producers and consumer. A circuit enables the control of data receivers based on information exchange over the network of connected senders.

In Figure 1 a cloud computing schema is illustrated. This model of DC requires three physical datacenters to create one virtual data center.

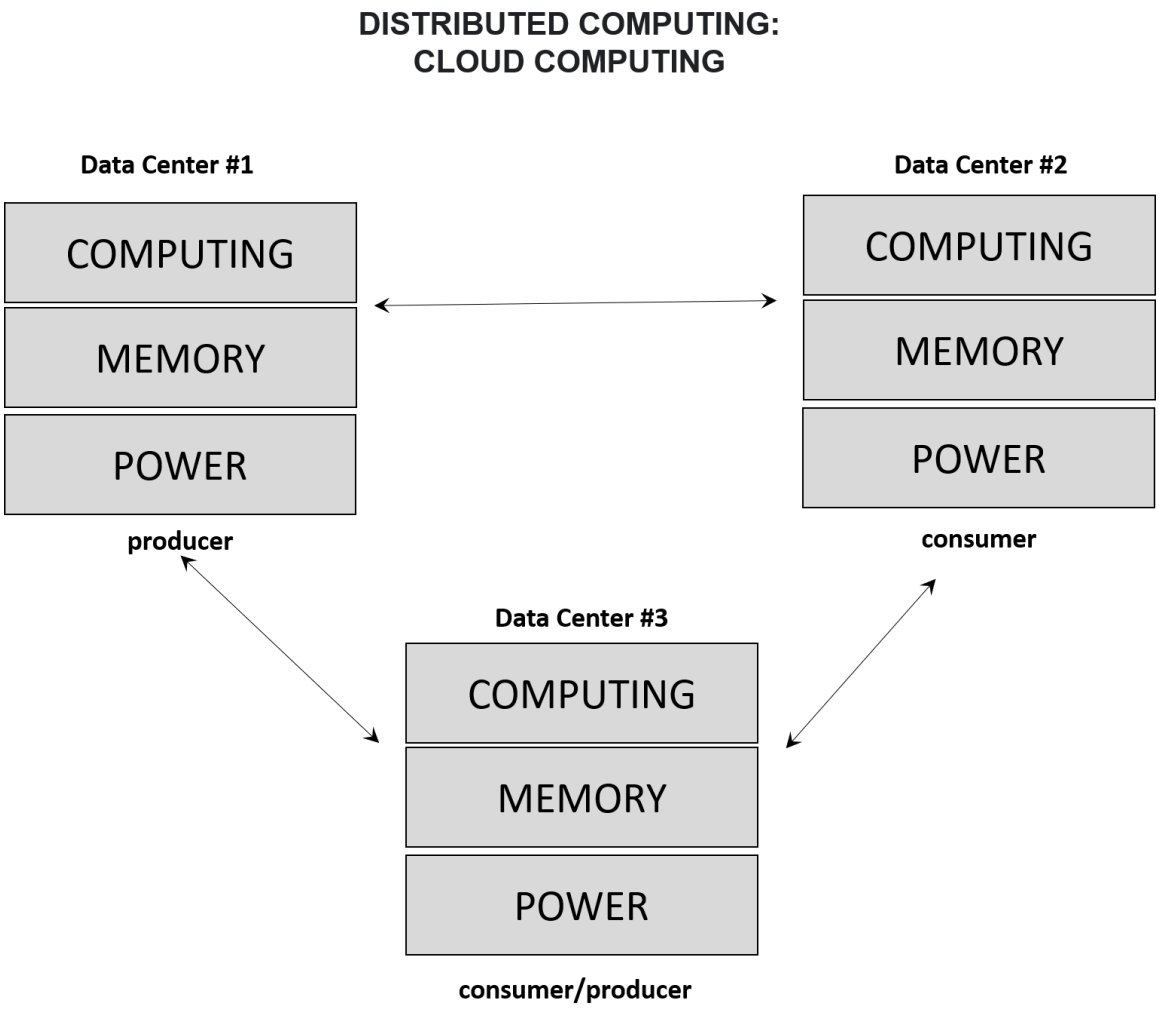

Figure 1 Minimal Cloud Computing infrastrucutre schema

Generally, the senders are storage units that communicate their information in specific media channel, according to the signals the storage units contain. The information routing and scheduling is an important optimization problem for many computer programs.

To communicate and describe information, a quantity of energy is required and is used to perform useful computational work (string entropy reduction) as coded by an arbitrary binary program schema and a set of random input strings. This value is the amount of energy transformed into work in the system. In addition, the energy source is also dissipated as lost to noise and heat, in the machine components.

The illustration of different patterns of energy consumption is shown in Figure 2. For example, a common pattern is the variation during day and night periods. During business hours the capacity utilization is at its peak for many commercial end-user applications such as with web applications. Other patterns observed are sporadic increase during marketing campaigns and monthly job batch processing.

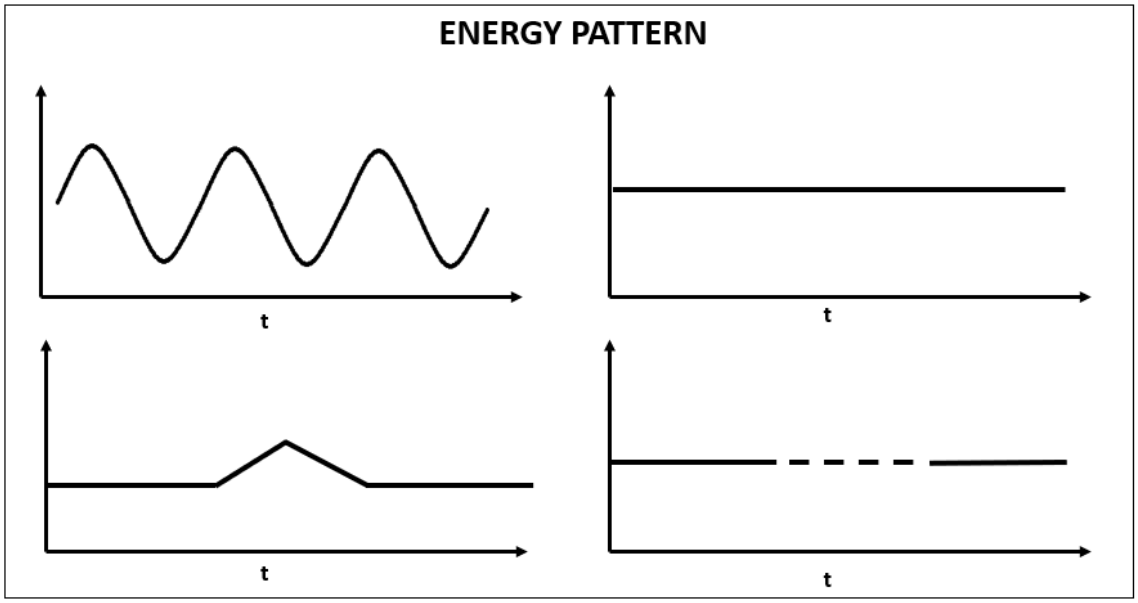

Figure 2 Types of energy usage for distributed computing

Overview

The complexity to compute any process is proportional to the search space of the problem. The most used metrics of performance concerns about the volume of memory and time required to complete a transaction. The mathematical description of computational complexity is undefined without the consideration of time.

The problem encoded is connected to the energy consumption in a machine such as proposed by Turing and Von Neumann, when instantiated as a Carnot Engine. This relationship is defined by an energy efficiency ratio measure in Joule per byte.

This is the maximum total number of bits that the machine can compute per Joule per bit per second encoded by a program. The computation rate is limited by the channel capacity of reading and writing a set of symbols from a finite language coded in a memory unit.

Therefore, the optimization for a problem-solving system can be described in terms of how to minimize the energy requirement to describe and compute a process that reduces entropy probabilistically between the regions of a random string, as described by seminal works from Shannon, Bernoulli, and Kolmogorov. This is the minimal energy necessary to successfully compute a bit of information.

An experiment was performed with the objective to maximize the output string of a utility function while optimizing the rate of energy change, under a degree of entropy variation.

With an iterative approximation approach and statistical modeling, the problem can be solved efficiently in terms of time and memory. The results show that the proposed computational model outperforms conventional heuristic benchmarks by better reducing entropy of an arbitrary set of random strings deployed in a physical system, for the same input parameters and similar code-length. Therefore, as we demonstrated in this paper, the statistical modeling of the output strings produces by a Bernoulli stochastic process allows a better energy efficiency as its produce's candidate strings with less entropy under less time and similar memory capacity.

The paper is structured in 6 parts: Introduction, Methods, Experiment, Results, Discussion & Conclusion.

Methods

Literature review

Hylton, Conte and Hill (Hylton, Conte, & Hill, 2021) have provided an insight on the impact of thermodynamic computation and how this model improves what can be achieved with classical computation models. In this sense they argue that all dynamic systems are limited by the law of thermodynamics and real-world scenarios such physical, chemistry and biological systems are naturally able to improve their functioning by dynamically adjusting the energy requirements. This is not a new concept and was previously analyzed independently in other works. For example, Benjegerdes (Benjegerdes, 2009) have presented in his seminar the same concept and argues that the relevant part

of a computation is the heat created during computation. This is the fundamental limitation in any practical computer and not the volume of operations per second processed by a real machine and a given input program.

Björnson and Larsson (Björnson & Larsson, 2018) have analyzed the performance of wireless networks in terms of its efficiency in bits per joule consumed for correctly receiving a bit of information. In the work they have modeled the upper limits in throughput in terms of energy consumption (bits/Joule) for a network with multiple antennas without interference, deployed in a spherical topology. They conclude that any data rate can be achieved by adjusting simultaneously the transmission power and bandwidth to an optimal ratio. They estimated the physical upper limit of energy efficiency is approximately 1Pbit/Joule.

Sun, Zhou and Hu (Sun, Zhou, & Hu, 2019) have proposed an algorithm to optimize the performance of networks of computer nodes such as in mobile edge computing, by integrating remote and local computation in terms of energy efficiency. In their model the Quality of Service (QoS) can be measured in terms of this metric. They proposed that properly allocating task based on energy and computational resources distributions, the system-level efficiency can be achieved. The conclusions found in the paper simulation suggest that the volume of data offloading to remote machines and local processing can have different impacts in terms of computational energy efficiency.

The approximate minimal computational complexity and the expected shortest string length of a program was modeled by Lima (Lima, 2021) in terms of statistical distribution of random sequence of output symbols produced by a heuristic Bernoulli process. In the work the approach was compared to a benchmark method and the simulation has provided better results in terms of memory and time. This has implications in terms of energy required to complete the execution and produce strings with lower entropy with less machine capacity utilization.

Information Description Problem. In the paper “Mathematical Theory of Communication”, Claude Shannon described information as a qualitative and quantitative model that is governed by a Bernoulli statistical process. Shannon have proved the following assertions:

The problem of communication of information occurs in three layers. For a message represented by discrete symbols encoded by a random source with a finite language and transmitted over a channel, there are 3 problems to address:

1. *How accurate the symbols are transmitted between input and output?*
2. *How precisely the messages are interpreted in the correct optimal meaning?*
3. *How effectively the meaning changes the behavior of the system?*

In this paper we propose a 4º problem:

How efficient we communicate and compute a program with the least amount of change in energy?

The *energy problem* is concerned to the effort to change the initial state function to the final state function. Mathematically this involves the challenges in programming a Universal Turing Machine to interpret and successfully compute one sequence of random Bernoulli variables. In Shannon mathematical model this is defined as a noisy channel.

Assuming the definition of "work" as the difference in state function output not lost to heat and noise by the machine. This can be quantified by the energy transfer rate. The efficiency is a concern of information complexity and is illustrated with the following question:

What is the optimal program that minimizes the amount of energy lost to heat and noise in a communication system?

In other words, we want to know how to maximize the amount of work being done by the machine to reduce entropy of a set of strings and thus providing information gain to the system.

Internal energy of a system. A system is comprised of a machine connected to an energy source, a finite set of logical programs and a list of strings of symbols. A program is a function computed by a machine with an input string. The exchange of signals between machine-program pairs forms a communication channel.

The internal energy of a system can be transformed (or converted) in two states as described in Table 2.

Derived State Function	Conversion Type	Description
Heat	Loss to thermal and quantum noise	Energy transferred to the motion of physical elements in a machine.
Work	Information gain to reduce entropy	Energy transferred to produce change in the state values of objects, such as binary strings, processed by the machine.

Table 2 Energy transformation in a machine

Particularly its difficult to know with certainty how much energy will be converted between noise and information by a *running* program.

The separation of the amount of Heat and Work being generated depends directly to the decision rules encoded in the process and the initial state of the program.

An instance of this dynamic is illustrated by the *hill climbing* technique used to optimize a state search. In this analogy the goal is to find the peak (maximum height) of a surface, where no neighbor symbol has a higher value.

The objective of the Hill climbing method is to optimize the information gain defined by the utility function used to evaluate the candidate string set.

Consider a program that starts with a random - arbitrary - initial solution and incrementally attempts to find a better candidate string that encodes a better solution. If the new value has a better alternative string, the program performs another interaction until no further improvements can be found.

Depending on the path chosen as the starting point, the program A will find only a local optimum. Another program B starting with a different initial state will have different decision outcomes and this will not necessarily be the optimal string (i.e., the global optimum) in the search space.

In the Figure 3 we illustrate the difference in performance efficiency in the hill climbing example:

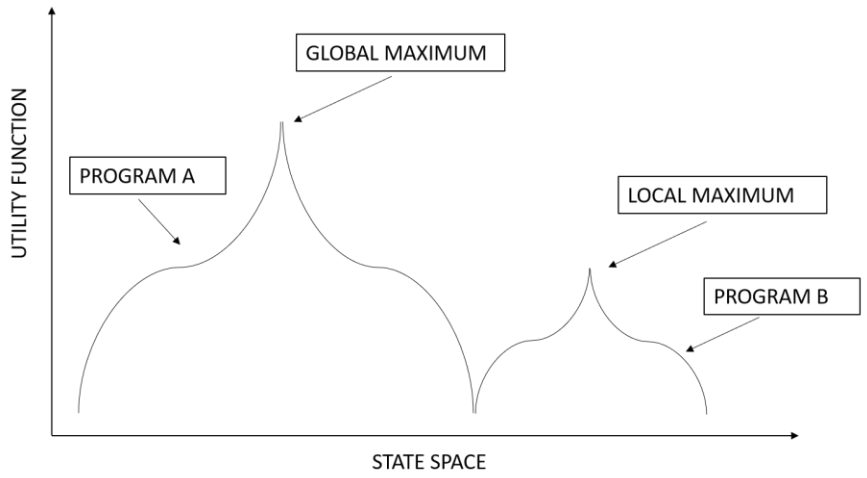

Figure 3 Global and Local states for a topology (Hill-climbing method)

Both programs transfer the same amount of energy, but one produces less noise than the other. Therefore, the amount of heat (information lost to noise) and work (information gain produced) depends on the decision made by the program during the computation operations (code) and the input strings provided by the source (i.e., The chosen pathway in the decision tree).

However, the change in energy is the same for program A and B. Thus, change in energy (i.e., Transition in a state function) is independent of the route used by either program.

The essential implication of this statement is concerned with the machine computing the program. Mathematically the most relevant part is the initial and final state. The intermediary states are important, but they introduce bias to the function schema and thus we tend to ignore them.

To measure the computation efficiency, we ideally would like to know exactly how much energy is required to a program to halt successfully. The problem is however, in many cases the search space is too large where the optimal string

could be encoded and, due to the non-computability of the Halting Problem, there are many pathways that leads to undecidable logical loops. Nonetheless, energy is a state function and thus we can model the change in energy.

To illustrate this, consider a problem modeled as a state space *S*, that is, a set of valid and invalid configurations a problem can be. In other words, S is a set of all possible states. Assume a program iterating though the elements *S* in each machine cycle. To illustrate assume for example an uninformed search method such as Depth-First Search (DFS) running for an arbitrary constant time *c*. We can't precisely know how much entropy has changed but we can say the value has changed and has decreased.

Similarly, we are concerned to the energy exchanged in the input and output interfaces of the system because of information being processed by the program-machine systems through its reading (the receiver unit) and writing (the sender unit) interfaces. In some instances, however we are only interested in the output state value being produced by the computation flow, regardless of the optimal level of entropy for the strings in the end of a process.

We can however use a state function derived on the rate of change between loss or gain of energy and entropy for nodes in a network. The formulation of enthalpy from thermodynamics is used to model this system.

Thermodynamics Concepts. Enthalpy is defined by the function

$$H = E + PV$$

Equation 1 Enthalpy equation

Where

- E is the internal energy of the machine
- P is the pressure
- V is the volume
- PV is the energy required to change the system

Assumptions

- E is irrelevant and is impossible to measure
- PV is the amount of the energy state function will change

We can re-write the formula of enthalpy as a change in values with the internal heat equation ΔE = q + w

$$\Delta H = q + w + \Delta\ (PV)$$

Equation 2 Enthalpy and heat equation

We introduce the two assumptions:

1. If P is constant than the determinant part is the volume
2. If the only work done in the system is the work w done to change the volume.

The formula can then be changed to

$$\Delta H = q + (-P\Delta V) + P\Delta V$$

Equation 3 Enthalpy with constant volume and pressure

A positive change (increase) in volume results in an output of work (information gain) by the system. A negative value is interpreted as a loss in the internal energy. Therefore, the sign of work and PΔV will be opposite and cancel each other. We can than conclude ΔH equals to the heat of the transmission. Change in enthalpy is equal to the heat gained or lost to the system. This energy is used by the system to reduce entropy.

$$\Delta H = q$$

Equation 4 Enthalpy is equal to the energy in the system

When a data transmission happens and enthalpy changes, the energy is transferred between the input and output of the system. The flow of energy is illustrated in the diagram in Figure 4

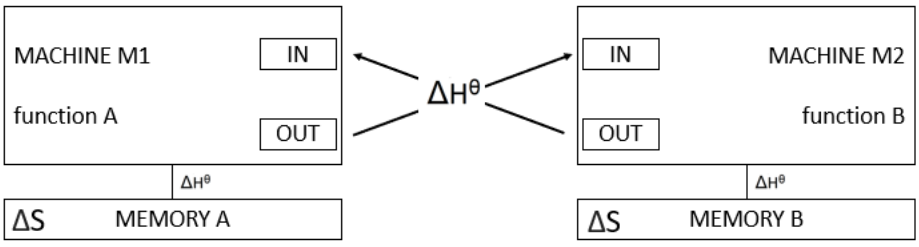

Figure 4 The exchange of bits (transactions) changes the enthalpy of the system and transfer the entropy between the machines tapes. M1 may be the same or different object than M2.

Data compression. The problem of data compression has implications on the amount of energy required by a program processing in a machine to reduce the coded information to a simple form, from a list of input strings. The compression method derives a language schema based on the observed input data. Some methods use the maximum-entropy probability distribution of symbols and words to infer this schema.

The model for this computation is illustrated in Figure 5

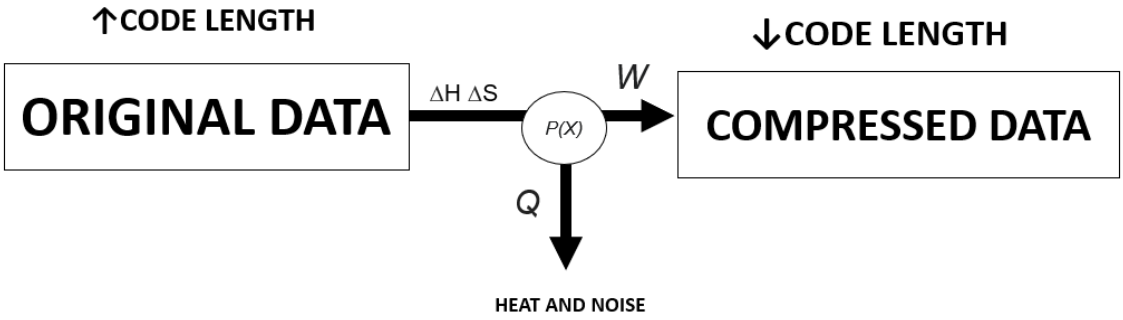

Figure 5 The machine requires energy to perform work of entropy reduction in a set of strings while also converting a proportion of this energy as heat.

Example. Assuming a random list of 255126 unique case-sensitive alphanumerical strings with length equals to 16 symbols for each coded word. The raw list has a length of 4337142 bytes (~4.1 M) and the compressed file has 3286973 bytes (~3.1 M). A subset of this list is shown:

BkTxLbTrpZES/PPA

WDm0hOE+B6A/Cy/n

KZXe3Exp1D3s48ko

I94SDPOpaG8F2cup

DrtgncIITB+QNc+b

eVMMVENvZs8qI4Fv

lSKXZ8tD9Vd1UnwG

wDCKNhWCfGsWfwms

Zbz5zX6Mls7tMyxR

The compression time for the GZIP algorithm is 0.293 seconds. Considering an energy requirement of 65W (joule per second), the processing had an energy cost of 19.045 joules.

The enthalpy change of the system can be measured. The added heat or lost can be calculated by measuring the temperature change of a known element such as silicon that is used to build a system. In a circuit the heat is transferring in and out of the atoms with the electrons moving between their orbitals.

From this we can determine how much useful energy is contained in a system. Using enthalpy, we can measure how much energy are in the bonds of the elements that compose the system. According to Hess's Law, the total enthalpy change of a transaction doesn't depend on the pathway used but only the initial and final states of the process.

If a process change can take difference paths whether in one or more interaction's, the amount of total heat change is the same, regardless of which branch is used to reach the final state.

The Hess's law argues that if a computation can have different valid strings representing more than one path and the initial and final conditions are the same, the overall enthalpy change is equal.

In other words, the enthalpy change for a thermodynamic process is the sum of the enthalpy change for each individual iteration, computed in a machine that instantiate the overall process. Therefore, if a set of strings can be grouped to instantiate the overall program, then their enthalpy changes can also be added.

Example: Consider a Language *L: {a,b}* with *X=P(a)* and *Y=P(b).* A string is a sequence of symbols s from L. Each string is a list of word that is distributed in an arbitrary order such as a stochastic process. There are two independent

sets A and B formed by a random combination of symbols in *L*, defined by *P(n,k).* The number of k-permutations of n signals in a channel. A program p is an instance of those coded rules being processed in a machine M. The energy change to complete this process can be calculated.

Example. A program p comprised of strings s_1 and s_2 computed by M has the overall state process:

$$[\ (1) \times a\ ] + [\ (3/2) \times bb\ ] \rightarrow (1) \times abbb$$

What is the enthalpy for the process in the system ΔH0 = ? kJ

Initial State:

$$[\ (1) \times a\ ] + [\ (1) \times bb\ ] \rightarrow (1) \times abb$$

$$\Delta H1 = -296.8\ kJ$$

Intermediate State:

$$[\ (1) \times abb\ ] + [\ (1/2) \times bb\ ] \rightarrow (1) \times abbb$$

$$\Delta H2 = -99.2\ kJ$$

The final string in the output is

$$[\ (1) \times a\ ] + [\ (3/2) \times bb\ ] \rightarrow (1) \times abbb$$

Then

$$\Delta H0 = -396\ kJ.$$

Assume information created by a working process in a discrete physical system moving entropy between two regions in a channel, with constant pressure and volume.

This process can be generalized for a discrete binary system with elements A and B

$$A = \{a(1), a(2), \ldots, a(n-1), a(n)\}$$

$$B = \{b(1), b(2), \ldots, b(n-1), b(n)\}$$

PV are pressure and volume in the system

m is the mass calculated as volume multiplied by density

Φ is the cumulative distribution function for the process

S is the entropy and H is the enthalpy of the system

The energy change ΔH schema during transactions is:

$$\Delta H = H + H'$$

$$H = \Phi(A,B) \times (S(A) \times m(A))$$

$$H' = \Phi(B,A) \times (S(B) \times m(B))$$

The relationship between A and B can be generalized:

$$Z = A \propto B$$

The quotient between the distribution of elements of A and B as rational numbers:

$$q(AB) = \text{antilog}\,(\log A(n) - \log B(n))$$

$$q(BA) = \text{antilog}\,(\log B(n) - \log A(n))$$

$$\Delta H = \sum(\Phi(\Delta q) \times S(\Delta Z) \times m(\Delta Z))$$

If mass is constant

$$\Delta H = \sum(\Phi(\Delta q) \times S(\Delta Z)))$$

The strings can be added and the enthalpy changes accordantly.

The machine computing this process is illustrated in Figure 6. The overall process occurs in a series of transactions with one or more paths.

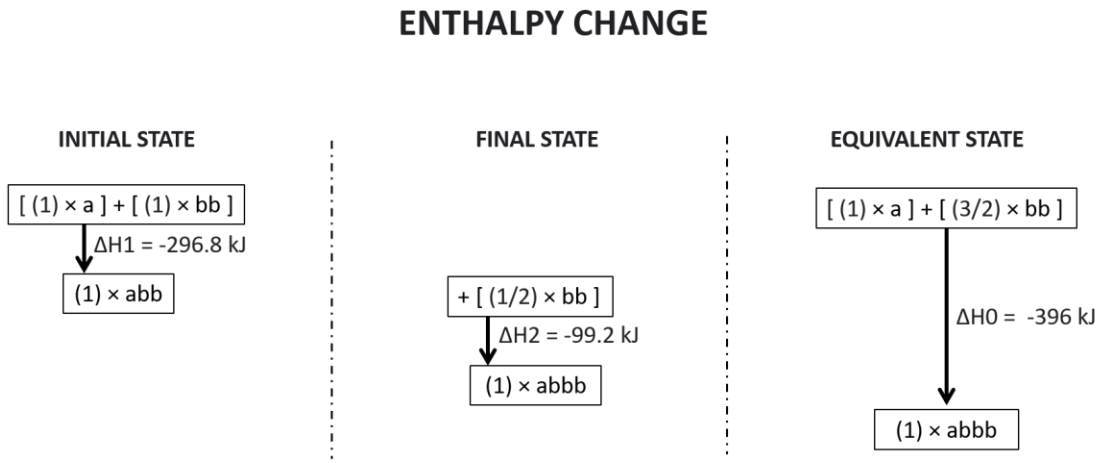

Figure 6 Change in the enthalpy for an overall process instantiating a program

This is the general schema for the overall signal processing during a finite number of cycles. The machine iterates by reading and writing words in the input and output interfaces, respectively. This operation is illustrated in Figure 7.

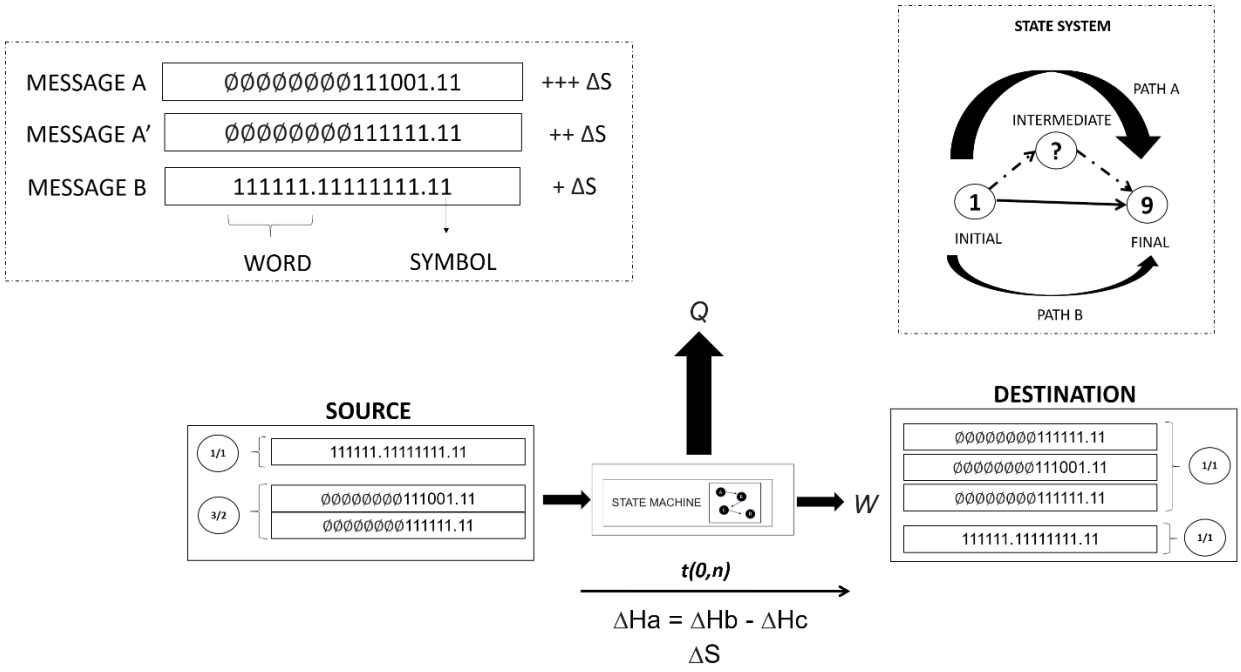

Figure 7 A thermo-computational model for a binary discrete system

The process analyzed must have the same standard environmental configurations. The system components must be constructed with a physical element with a standard enthalpy of formation. The machine must behave in a predictable operation specified by a state function with a discrete Language, a Grammar rule coded in a program, regardless of the complexity. The enthalpy changes for a transaction in the machine can be calculated even if it's not possible to measure directly. Hess's law is defined with the formula

$$\Delta H_{net} = \sum \Delta H_t$$

Equation 5 Hess's Law

Where

- ΔH_{net} *is the net change in enthalpy*
- $\sum \Delta H_t$ *is the sum change in enthalpy of transactions*

A system can be in different entropy phases (high, medium, low) proportional to the change in enthalpy. This is illustrated in Figure 8

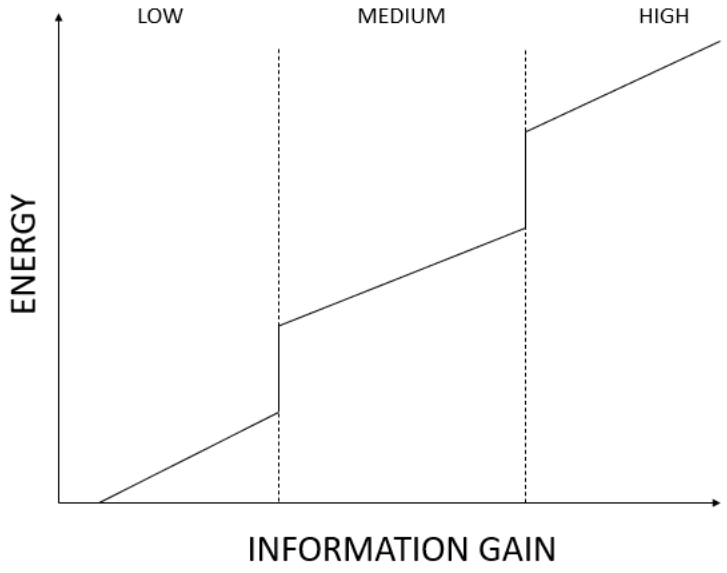

Figure 8 Relationship between information gain and energy requirement for a system

As an example, lets calculate the heat change by a program being processed in a machine. First, we must find the mass of silicon's that constitute the transistor used to build the machine.

Example. Determinate the amount in grams of Silicon (Si) are in a computer processor with x = 1.08 * 10^{23} atoms of Si. The Silicon has 28.09 g/mol. The Avogadro's number is 1 mole = 6.023 * 10^{23} particles.

The amount of moles can be calculated with the equation:

$$\textit{Number of moles } x = \textit{Number of atoms of the sample } y \div \textit{ Avogadro number}$$

Equation 6 Mass equation

Therefore, the moles of the silicon are $x = (1.08 * 10^{23} / 6.023 * 10^{23}) = 0.18$ *mol*. If 1 mole of Si weights *28.09g* and *0.18* moles contains *(28.09/1) * 0.18 = 5.04 grams (mass)*

Heat equation. Now let's calculate the heat change in a transaction being computed in this system.

From the calorimetry formula defined as

$$Q = m \times c \times \Delta T$$

Equation 7 Heat equation

With parameter:

- Q is the enthalpy or heat change in Joules
- m is the mass of the system in grams
- c is the specific heat capacity of the element
- ΔT is the change in the temperature of the system

Example. Consider that the mass of silicon in a machine processor is 5.04 g. This machine is computing an arbitrary program during a period (cycle). The initial temperature of a processor during the transaction is 50°C and 62°C under full load. The added or lost energy in the system can be calculated by measuring the temperature change of an element caused by the process execution. Therefore, we can determine the heat change of the system.

The specific heat capacity *c* of Silicon (Si) is $0.7\ J\ g^{-1}°C^{-1}$

Temperature change in the system $\Delta T = (62 – 50)\ °C = 12\ °C$

Heat change $Q = m \times c \times \Delta T$

$$Q = (5.04\ g) * (0.7\ J\ g^{-1}°C^{-1}) * 12\ °C$$

$$Q = 42.336\ J = 0.0423\ kJ$$

Turing Machines as Carnot Engines. A Turing Machine M can be implemented as a *Carnot heat engine* with its internal logic (truth table) being the mechanisms of the *Carnot cycle*. M can be in many different states in the space set. The variation between states is the thermodynamic cycle. As the system changes it produces work and heat. The machine transfer energy between storage space of higher and lower entropy. This process can be reversed transforming a storage with lower entropy to a higher entropy.

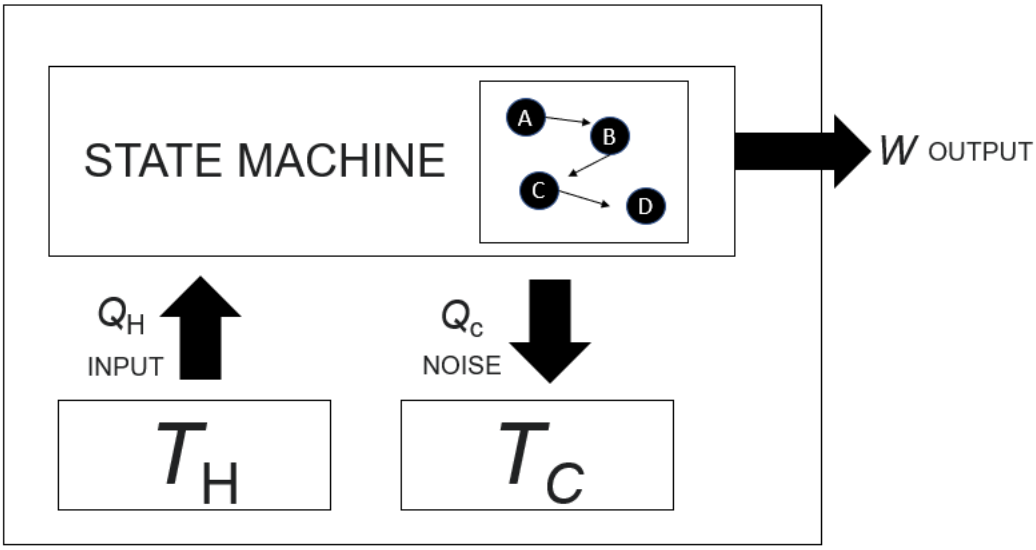

Figure 9 Turing Machine idealized as a Carnot engine

The Figure 9 illustrate a Turing Machine idealized as a Carnot engine. The amount of heat Q_H is transferred from a high entropy space through the circuit T_H. The remain heat Q_C is transferred to the low entropy storage T_C. This flow of data produces work (information gain) in the system through the cycles of a machine-program system interaction.

The movement of energy and entropy in a network of machines for an overall processing cycle is represented in Figure 10. The system can be built with one or more machines. The machine has two parts, the sender, and the receiver units. The sender object produces information work by moving strings from high entropy region to a lower state receiving part. As a Carnot engine, the system processing information also converts energy to heat with a proportional work-to-noise ratio.

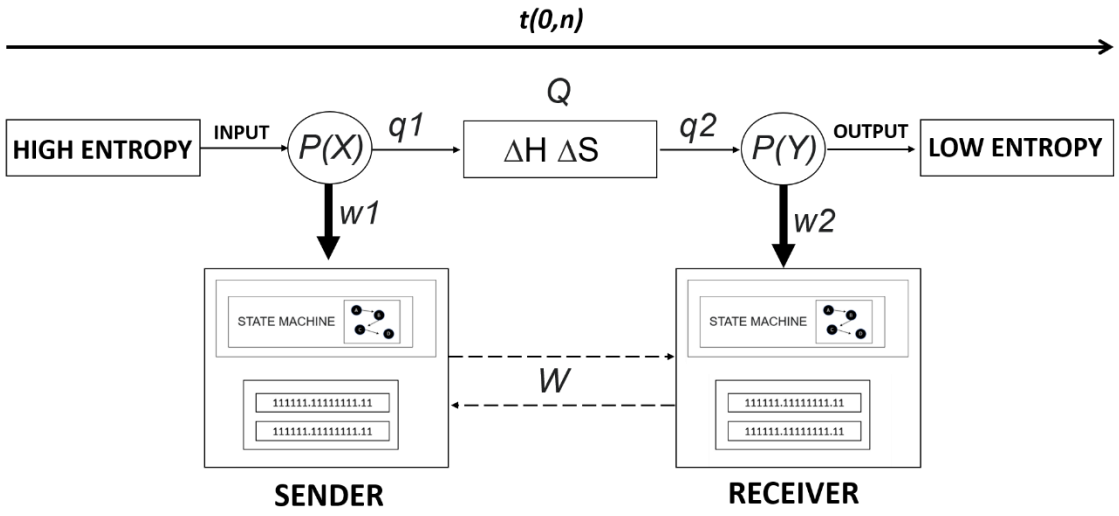

Figure 10 A thermo-computational system processing a set of strings

Types of process. To understating the computation of information in heat engines we must describe the difference in the behavior of process. There are two types of thermodynamic process: Reversible and Irreversible.

For a Reversible (Ideal model) it has the following characteristics:

- Process in which both the machine and the program can be simultaneous returned to the initial state after the process has concluded
- There is an equilibrium in the state machine between initial and final states of the system
- Infinite state changes occur in the system

For an Irreversible process:

- In this case the system original state function cannot be returned when completion.
- Finite changes occur in the system
- There is no equilibrium in the system

The Figure 11 illustrates the difference between process in a state graph.

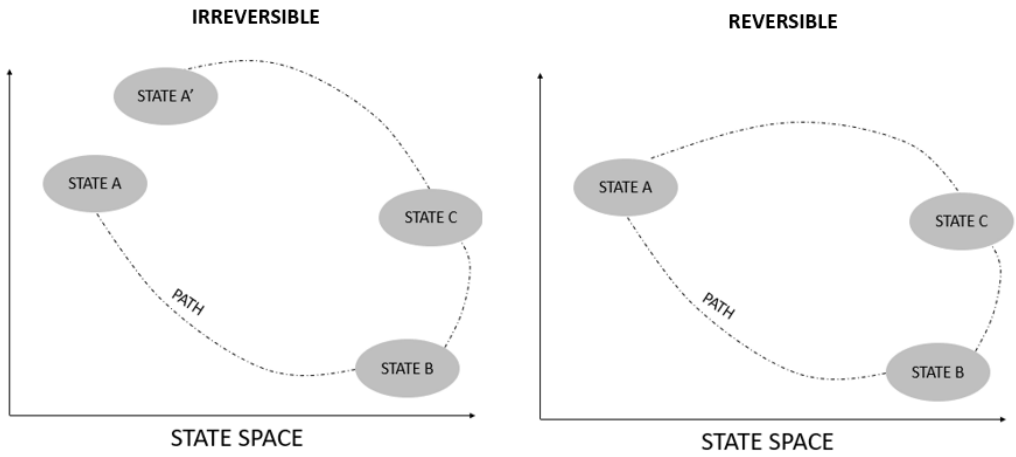

Figure 11 Representation of types of programs states computed by a machine

When the system moves between states its entropy change is the same (ΔS) for both a reversible and an irreversible process.

The total change in entropy in a reversible process is equal to zero. The entropy of individual interactions can change but the total change in entropy of the system is zero. The system does not change the entropy of the internal of a system and its interfaces with external systems.

The entropy increases for any system in an irreversible process. Entropy is constant for reversible process however it increases for an irreversible process. The entropy change for a theoretical reversible process is the same of an irreversible process between initial and final state.

Example. A machine processes a program that reads and writes signals in a string continually. The temperature of the machine is 500 K. Estimate the entropy change if the program executes by two hours with a power input supply of 30 W. Consider a reversible process such that

$$\Delta S = \int (dQ / T)$$

$$\Delta S = Q / T$$

$$\Delta S = (0.03 * 2 * (3600))/500$$

$$\Delta S = 216 / 500 = 0.432\ kJ$$

Example. Calculate the entropy change in 1 gram of Silicon (Si) at 30 °C to 80 °C with constant pressure. The specific heat of Si is 0.7 J $g^{-1}°C^{-1}$ than,

$$\Delta S = C_p \ln (T2/T1) = 0.7 * \ln ((273+80) / (273+30))$$

$$= 0.7 * \ln (353/303)$$

$$= 0.7 * \ln (1.1650)$$

$$= 0.1527 * 0.7 = 0.106 \text{ kJ/Kg K}$$

Measuring Efficiency. The efficiency of the machine is defined by

$$\eta = \text{Output Work / Input Energy}$$

Equation 8 Energy Efficiency

Consider the parameters:

- T_H is the Source entropy storage
- T_C is the Destination entropy storage,
- Q_H is the energy flow from Source to the Machine M input
- Q_C is the energy lost to noise as heat (i.e., Thermal and Quantum Noise)
- W is the work output that produces information gain

W is the energy balance difference with formula

$$W = Q_H - Q_C$$

Equation 9 Energy balance

Then the efficiency formula is defined in terms of Q and W:

$$\eta = W / Q_H$$

$$\eta = (Q_H - Q_C) / Q_H$$

$$\eta = 1 - (Q_H / Q_C)$$

Measuring Noise. Noise can be informally generalized as an undesired signal oscillation transmitted through a channel. In electric systems there are two important types of noise: Thermal Noise and Quantum Noise.

Thermal noise was first described by John Bertrand Johnson and Harry Nyquist when working to explaining the basic source of random interference in communication channels. This type of electronic noise is randomly generated by the thermal oscillations of the electrons in a circuit due to variations in the charge. This affects all electric circuits and communications senders and receivers and defines a limit in the throughput of a process. The level of noise increases with temperature. For a finite bandwidth it can be modeled as a Normal distribution. The collision created by the vibration of atoms and electrons in the circuit causes a random motion and thus increases the degree of noise in the system. The rms noise is defined as:

$$V = \sqrt{(4kTRB)}$$

Equation 10 Thermal noise equation

Where

- V is the voltage of the noise
- K is the Boltzmann's constant
- T is the temperature (Kelvin)
- R is the resistance (Ohms)
- B is the bandwidth of the system (Hertz)

Example. In a MOSFET transistor the Drain to Source ON-Resistance ($R_{DS(on)}$) is the resistance between the drain and source components when a specific Voltage Gate-To-Source (V_{GS}) is applied. When V_{GS} increases the $R_{DS(on)}$ decreases. The transistor works with 30V and 0.56A. At 4.0V the resistance is 1.0 Ω and at 2.5V is 1.5 Ω. An Integrated Circuit (IC) have between 10 to 100 transistors.

A processor works with a temperature of 60°C (333.15 K) and a bandwidth frequency of 4.40 GHz. Assuming a transistor resistance of 1.5 Ω in the

circuit, we can determine the generated noise voltage. Let's calculate the noise power:

$$Boltzmann's\ constant\ k = 1.38 * 10^{-23}$$

$$1\ Mhz = 10^{6}\ Hz$$

$$1\ GHz = 10^{9} Hz$$

$$Noise\ Power\ P_n = k * T * B$$

Where

- P_n *is the noise power (Watts)*
- *k is the Boltzmann's constant (Joules per kelvin)*
- *B is the bandwidth (Herts)*
- *T is the temperature (Kelvins)*

$$P_n = (1.38 * 10^{-23}) * (333.15) * (4.4 * 10^{9})$$

$$P_n = 2.0228868 \times 10^{-11} W$$

The noise voltage is defined by the formula:

$$V_{rms} = \sqrt{(4kTRB)}$$

$$V_{rms} = \sqrt{(4 * k * 333.15 * 1.5 * 10^{9})}$$

$$V_{rms} = 5.25336262\ \mu V\ RMS\ (-83.4\ dBm)$$

Quantum noise is the uncertainty of a physical quantity that consist of discrete quantized packets. It's a random process that occurs because of variations in the number of electrons between source and detectors in a circuit. It's the fundamental limit possible in a signal-to-noise communication system. Although the average flow may be constant, there are more energy received at a given moment than in another instant. The concept of quantum noise was described by Walter Schottky. The formula for quantum noise is described bellow

$$I_{rms} = \sqrt{(2qIB)}$$

Equation 11 Quantum noise equation

Where

- q is the electron charge ($1.602 * 10^{-19}$ C)
- I is the current flowing in the system in the received interface (Amperes)
- B is the bandwidth (Hertz)

Example. Consider the current present in a circuit with 0.56 A flowing across a semiconductor and a noise bandwidth of 1 Mhz. The quantum noise can be determined:

$$I_{rms} = \sqrt{(2 * (1.602 * 10^{-19}) * (5.6 * 10^{-1}) * 10^{6})}$$
$$I_{rms} = \sqrt{(17.9424 * 10^{-14})}$$

Experiment

Machine structure. To perform the experiment implementation, we have considered a machine with two components as describe in Table 3.

Layer	Measure	Category	Description	Example

Computing	Proportion of used cycles per second	Information Processing	Conceptualized as the control head in the Turing Machine	Logical Arithmetic Unit (CPU)
Storage	Bits	Information Representation	Conceptualized as a tape in the Turing Machine	• Local Memory Array • External network channels

Table 3 Machine model description

A process computed in a system of one or more machines is illustrated in Figure 12.

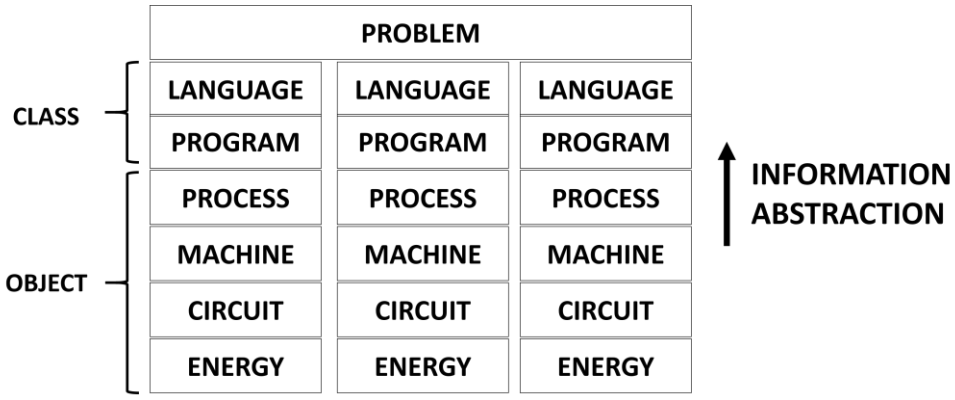

Figure 12 The levels of abstraction from 1. a class of problem description 2. to the object instantiation in a circuit network. This system computes an arbitrary process powered by an energy source.

The programs are loaded in the CPU registers and the bits described are transferred between locally internal and external storage systems. The units of the Storage and Network Layers are measure in Bits. The units of the Computing Processing Layer are measured as a percentual proportional to the utilization rate per second by a program in a time sequence to complete a full cycle and it's defined with a domain D between [0,1]. The rate of processing and transmission per cycle is:

$$R_{(D,\,C)} = Data\ Layer\ /\ Computing\ Layer$$

This function valued is measured in bits per cycle.

Experiment Description. To measure the energy consumption change, we have devised an experiment to calculate the number of joules per bits required by a program coding a heuristic approximation. The non-polynomial asymptotic function is defined as:

$$f(x) \sim g(x) \text{ when } x \to \infty$$

$$\text{if } \lim_{x \to \infty} f(x) / g(x) = 1$$

Equation 12 Asymptotic function definition

Traveling Salesman Problem. The loss to heat is concerned to the amount of energy that is not converted to generate gains of information. This has implications in many engineering challenges if one thinks of computation and communication in general. Consider, for example, the effort of a program to compute the routing optimization for the Messenger/Traveling Salesman Problem from William Hamilton. In broader terms it asserts:

- A path or route is a description string that quantify the relative logical relationship between symbols such as the Euclidian distance for a set of variables in an arbitrary geometric topology.
- For a list of cities, for example in a 2-dimensional cartesian nodes, what is the shortest path that visits every city only once and returns to the first city in the end?

In those problems the salesman-messenger is an actor with a decision-making role over the information associated to an arbitrary state function.

There are two variations (or special cases) to model this computing problem:

- *Infinite resource agent*, whereas the agent can evaluate the reference string and alternative random candidate indefinitely in the search for the global maximum, as the memory and time are considered virtually infinite resources.
- *Finite resource agent,* the agent must decide when to halt computation using only the know observed information and considering a scarce, finite energy resource capacity. This is related to the concept of *gamblers*

ruin from statistics in the sense that the agent playing must avoid consume all resources and lose all wealth, “go broke”. This can be achieved with fractional betting using side information in the channel to maximize the growth of the utility function associated to the problem, under a given expected entropy threshold as estimated by the Kelly criterion.

The following assumptions were made to model the problem (TSP). The language to represent a city has a 2-dimensional coordinate system:

- L_1: *{(0-9): ([0-9], [0-9])}*
- L_2: *{A_m: p(x,y)}*
- L_3: *{A: (0,0)}*
- L_4: *{B: (2, 3)}*
- L_5: *{C: (5,6)}*

A solution string is formed by a grammar rule G. A sentence, also called a string, is a sequence of symbols in a Language L with a given G. This system can be generalized in such a schema that

$$G: [A_{m=1,} A_{m=2,} A_{m=3,} A_{m=4,} A_{m=n-1,} A_{m=n}]$$

Where A_m is not equal to any other A_n

The combination of symbols in different orders directly affects the information gain obtained by the utility function, in this problem this is the Euclidean distance. Other NP problems have different schematics associated but all of them address problems in a large-scale search space volume. For example, the associated cost function G’, G’ and G’’’ for a random solution string:

G’: [1, 2, 5, 4, 3] = 100 distance units (cm, meters, etc.)

G’’: [4, 2, 5, 1, 3] = 110 distance units

G’’’: [4, 2, 1, 5, 3] = 80 distance units

The objective of this method is to find the string with the smallest total distance. The utility function is the Euclidean distance in a 2-dimensional plane. For two points: $d = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$

In this model the program and the input strings are re-arranged in each sequence order by a Universal General Machine. The output strings values follows a log-normal distribution.

The input parameter data has 280 words. A word encodes a node in the graph such as A_m: $p(x,y)$. One for each city in the TSP instance used in the experiment. For example, in a 2-dimensional topology:

- *1 288 149*
- *2 288 129*
- *3 270 133*
- *4 256 141*
- *5 256 157*
- *…*
- *278 252 125*
- *279 260 129*
- *280 280 133*

Candidate strings are generated arbitrarily by randomly flushing the elements in the string created from the language schema programed in the machine memory. Examples of valid and invalid strings schemas:

- *Valid*
  - *L': {A, B, C, D, E} -> {B, D, A, E, C}*
  - *L': {A, B, C, D, E} -> {D, B, E, A, C}*
- *Invalid*
  - *L': {A, B, C, D, E} -> {A,A,B,C,D,E}*
  - *L': {A, B, C, D, E} -> {A ,E}*

Alternative strings that produce better results (information gain) are chosen iteratively as the approximated optimal string. Solutions strings with a worst utility function state are accepted according to a decaying probabilistic function that simulates the optimization in entropy flow for a machine computing a finite Bernoulli process. The algorithm is demonstrated in Figure 13.

QA method

- initial string S= random permutation ({D, B, E, A, C})
- expected optimal string S' = initial string
- expected cost C_{best} = utility function (expected optimal string)
- decrement rate d_r= 0.01
- information gain odds b=1
- initial probability P_r (1) = 1.0

- For i from 0 to maximum integer number of interactions N
    - alternative string = random permutation (expected optimal string)
    - cost $C_{alternative}$ = utility function (alternative string)
    - If $C_{best} < C_{alternative}$
        - expected optimal string S = alternative string S'
        - $C_{best} = C_{alternative}$
    - else
        - X = decaying Kelly criterion probabilistic function (P_r(1), d_r, b, i)
        - Y = random Bernoulli process with constant P_B (1 = 1/100)
        - If statement (X OR Y) is True
            - expected optimal string S = alternative string S'
            - $C_{best} = C_{alternative}$
        - P_r(1) = Pr(1) – decrement rate

- write (expected optimal string S)
- write (expected optimal cost C_{best})

Figure 13 Quantitative Algorithm

The simulation was performed as a trial with length l=15 for each algorithm with a total sample size of 30. Each word in the problem is represented by a numerical Language L with integer values. For each word is assigned a respective city-node with an associated X and Y coordinates in a graph

topology. A sequence of words codes a path in the graph. For example, L': [5: (256, 157), 1: (288,149), 2: (288,129) … 280: (280,133), 279: (260, 129)].

During a full process each method executes for a fixed number of computer cycles t=300. This number was chosen to satisfy a given degree of freedom. This is the amount of allowed entropy change in a dynamic system.

The number of machine cycles used to compute the entire process in the experiment with two programs (p=2) is

$$M_{computed\ cycles} = p*((l*t) + c)$$

Equation 13 Number of cycles in the experiment

Where c is a constant time before entering the loop statements in the program and performing data copy, assignment and swap operations that occurs in $O(c)$ for a given string.

The machine computes a given encoded program, a random input data and parameters. This combination is the system where the transactions event occurs. The processing of transactions generates work but also heat and noise. This increases the temperature and energy consumption in the machine. The computational capacity is limited by the heat interference in the processor and storage unit within the circuit.

The Shannon-Hartley theorem defines the maximum rate at which information can be transmitted over a (Gaussian) noisy channel. The channel is a continuous-time analog communications system with a given bandwidth. This theorem defines the maximum channel capacity for an error-free signal transmitted with a specific bandwidth with interference.

Stochastic problems. The proposed TSP instance assumes some statistical properties and is applied to special cases for stochastic decision problems. By considering the problem class solution as a process optimization for less restricted entropy variance:

- The strings symbols have a signal and noise behavior that is locally stationary, but is also globally nonstationary such with a generalized Wiener process
- The probability for each symbol is a Bernoulli variable X with equal probability for all symbols in a language

- The language L is finite and can be computed by a given finite state machine with an input string s, in a discrete system
- The computing machine is modeled as a Turing Machine M
- The level of uncertainty over possible function state values is calculated with the Shannon entropy S(X)
- The program coding the problem have a log-normal dependent utility function $U=\min\{f(X_n)\}$ for n permutations of the input string
- This system is a generalized ergodicity process
- The solution strings are coded as a sequence of random permutation of unique symbols
- The utility function has a real value ranging from 1 to infinite and cannot be negative
- The machine requires a non-zero quanta of energy to describe and compute the programed information
- The energy in the system is converted in work to optimize the strings regions
- Work is a function to reduce entropy between the random variables processed by the system
- The energy is also lost to heat and noise during all cycles according to a thermodynamic process
- The physical system is an instance of the Carnot Engine M_c

Computational performance. The performance of the machine-program system can be measured as described below:

- L is the load average, is the average system load over a period such as 1, 5, and 15 minutes.
- P is the CPU idle time for the machine utilization rate. This is percentual of time that the machine "wasted" without performing useful instructions relevant to a program.

The algorithm used in the experiment was the Quantitative Algorithm (QA) and tested against the results from the Simulated Annealing (SA) method, the benchmark algorithm. The QA algorithm method requires less computational resources such as string operations than traditional algorithms. This is observed for nature-inspired methods like Genetic Algorithm, Simulated Annealing and Convolutional Neural Network.

The proposed method does not have the same pre-defined bias in the code program such as in traditional methods, as it uses a dependent decreasing probabilistic distribution of a simulated utility function assigned to the problem. For example, this can be observed for a Bernoulli process derived on the Euclidean distance over a given topology.

The method analyses the statistical behavior about the string output values to decide when to stop computation. The QA method has a better performance than SA as its produces results with less time, similar number of operations and code length per coded word, and more solution quality. The QA model has results measured with tested statistical significance as it has a *p-value* less than 0.05 for the observed data. Therefore, it obtains optimized likelihood and string improvement rates than other heuristics approximation methods.

To remove the background noise and ensure a fair evaluation we have measured the performance of the machine without the program being computed. With that we want to quantify the amount of effort that occurs independently of the experiment execution and minimize any bias in the results. If any program was competing for resource capacity in the machine, we could measure the impact the machine was spending making context switch when swapping process between the CPU and the scheduling queue.

Measured in a period of 10 minutes, the machine had a load average L_{avg} prior to execution of the experiment of 0.02 and a CPU idle time P_{idle} of 99.9%.

The machine used in the experiment has the following capacity:

- The temperature change in a processor is 50°C to 62°C.
- The average power requirement for a processor is 65 W to 95 W during full load capacity.

- The experiment was computed by a machine with 1 virtual CPU (i.e., a shared resource access to physical central processing units) and 2048 MB of memory (RAM).

From the experiment execution we have recorded the following observations.

In Table 4 we have the machine capacity in Watts and Temperature range.

Machine Capacity	Value
Max Temperature (Kelvin)	335.15
Min Temperature (Kelvin)	323.15
Max Power (Watts)	95
Min Power (Watts)	65

Table 4 Machine capacity

The load averages are shown in Table 5

Load Average	Before	After
1 Min	0	0.21
5 Min	0.02	0.48
15 Min	0.02	0.29

Table 5 Load average

The program parameters are described in Table 6

Program Parameters	Value
Words in Language	280
Number of Interaction	300

Number of Computing Cycles	9000
Symbol in Language	10
Input Data Length (Byte)	1441
Sample Size per method	15
Quantity of computed programs	2

Table 6 Program parameters

Results

In Table 7 we compare the program and input length, volume of bytes per word, average initial state cost, average optimal cost, average string improvement rate, average computation time and average expected energy for QA and SA methods.

Function	**Program Code Length (Byte) - Compressed**	**Bytes per word**	**Average Initial State**	**Average Optimal Solution**	**Average String Improvement**	**Average Computation Time**	**Average Expected Energy**
QA	1398	0.200286123	34309.03176	30320.60745	11.52%	25.68074738	0.7316
SA	1367	0.219458669	34266.99576	31303.90213	8.53%	28.25617467	0.7962

Table 7 Experiment results

In Table 8 and

# Execution	Heuristic Function	Initial State Cost (float)	Final State Cost (float)	Percentage Change (Information Gain)	Computing Time (Seconds)	Expected Energy (J/Byte)
1	SA	35495.893	31165.9656	0.1220	27.49475966	0.7748
2	SA	33161.944	31420.2838	0.0525	28.85200331	0.8130
3	SA	34474.095	31745.4089	0.0792	28.34268424	0.7987
4	SA	35251.781	31418.3977	0.1087	27.41144393	0.7724
5	SA	33625.155	31369.467	0.0671	28.55964353	0.8048
6	SA	32169.345	30983.9806	0.0368	27.84305172	0.7846
7	SA	35497.444	31420.1086	0.1149	28.07026922	0.7910
8	SA	32106.127	31258.9379	0.0264	27.36997414	0.7713
9	SA	34484.886	31394.4589	0.0896	28.13790775	0.7929
10	SA	34165.051	31304.4588	0.0837	28.59434832	0.8058
11	SA	32850.746	31340.9377	0.0460	28.91900824	0.8149
12	SA	36525.999	31655.0848	0.1334	28.81544189	0.8120
13	SA	34388.33	31301.6523	0.0898	28.75189129	0.8102
14	SA	34450.735	31198.7993	0.0944	28.09958007	0.7918
15	SA	35357.406	30580.5898	0.1351	28.5806127	0.8054

Table 9 we provide the experiment results for the execution of QA and SA approximations.

# Execution	Heuristic Function	Initial State Cost (float)	Final State Cost (float)	Percentage Change (Information Gain)	Computing Time (Seconds)	Expected Energy (J/Byte)
1	QA	33028.631	30834.9953	0.0664	25.68601025	0.7318
2	QA	36890.453	30374.5548	0.1766	25.97841623	0.7401
3	QA	35769.022	29901.351	0.1640	25.10210514	0.7152
4	QA	33509.294	30009.4833	0.1044	26.33352285	0.7502
5	QA	33965.079	30078.5758	0.1144	26.55812006	0.7566
6	QA	33753.861	30542.9283	0.0951	24.95599793	0.7110
7	QA	34552.545	30368.345	0.1211	25.766705	0.7341
8	QA	35652.537	30343.2864	0.1489	25.22375764	0.7186
9	QA	34417.638	30318.8124	0.1191	25.7886909	0.7347
10	QA	33571.665	30793.4611	0.0828	25.46285851	0.7254
11	QA	35510.453	30185.4876	0.1500	26.11169088	0.7439
12	QA	32486.746	30011.6777	0.0762	24.92256997	0.7100
13	QA	33651.855	30450.3156	0.0951	25.89155718	0.7377
14	QA	34500.613	29474.2329	0.1457	25.50205487	0.7266
15	QA	33375.084	31121.6046	0.0675	25.92715328	0.7387

Table 8 QA execution results

# Execution	Heuristic Function	Initial State Cost (float)	Final State Cost (float)	Percentage Change (Information Gain)	Computing Time (Seconds)	Expected Energy (J/Byte)
1	SA	35495.893	31165.9656	0.1220	27.49475966	0.7748
2	SA	33161.944	31420.2838	0.0525	28.85200331	0.8130
3	SA	34474.095	31745.4089	0.0792	28.34268424	0.7987
4	SA	35251.781	31418.3977	0.1087	27.41144393	0.7724
5	SA	33625.155	31369.467	0.0671	28.55964353	0.8048
6	SA	32169.345	30983.9806	0.0368	27.84305172	0.7846
7	SA	35497.444	31420.1086	0.1149	28.07026922	0.7910
8	SA	32106.127	31258.9379	0.0264	27.36997414	0.7713
9	SA	34484.886	31394.4589	0.0896	28.13790775	0.7929
10	SA	34165.051	31304.4588	0.0837	28.59434832	0.8058
11	SA	32850.746	31340.9377	0.0460	28.91900824	0.8149
12	SA	36525.999	31655.0848	0.1334	28.81544189	0.8120
13	SA	34388.33	31301.6523	0.0898	28.75189129	0.8102
14	SA	34450.735	31198.7993	0.0944	28.09958007	0.7918
15	SA	35357.406	30580.5898	0.1351	28.5806127	0.8054

Table 9 SA execution results

- QA has an average string improvement rate of 11.52% and SA has 8.53%. The average time to completion for QA and SA is 25.6807 and 28.2561 respectively.
- The proposed program QA has a smaller power consumption per bits than the benchmark heuristic with an average of 0.7316 J/byte (QA) and 0.7962 J/byte (SA)
- QA has a better optimal string candidate than SA. QA has an associated utility cost of 30320.60745 versus 31303.90213 for SA.
- The average initial value for QA and SA is 34309.03176 and 34266.99576, respectively.
- QA has a ratio of Bytes per Word of 0.200286123. SA has a ratio of 0.219458669

The expected energy (in Joule per bytes) in the machine is defined by the formula:

$$M_{Q/S(Joule/byte)} = (t * ((\ Q_{max} + Q_{min}\ ) / 2\ )) \div (\ P_{Length} + In_{Length}\ )$$

$$M_{Q/S(Joule/byte)} = (t * E(Q)) \div \sum S$$

Equation 14 Machine energy requirement in Joule per byte

Where

- t is the running time for the program (seconds)
- Q is the energy used by the machine (Joules per second)
- E(Q) is the expected energy change in the system
- P_{Length} is the code length of the program (bytes)
- In_{Length} is the input data length (bytes)
- $\sum S$ is the sum in length of all strings in the system

The minimal quantity of information gain is equal to a process successfully sending 1 bit of information for 1 second in a channel connecting strings with different entropies. This is interpreted as a unit of computational work.

In Figure 14 we have the CPU utilization rate measured as a function of the idle time in the processor during the experiment.

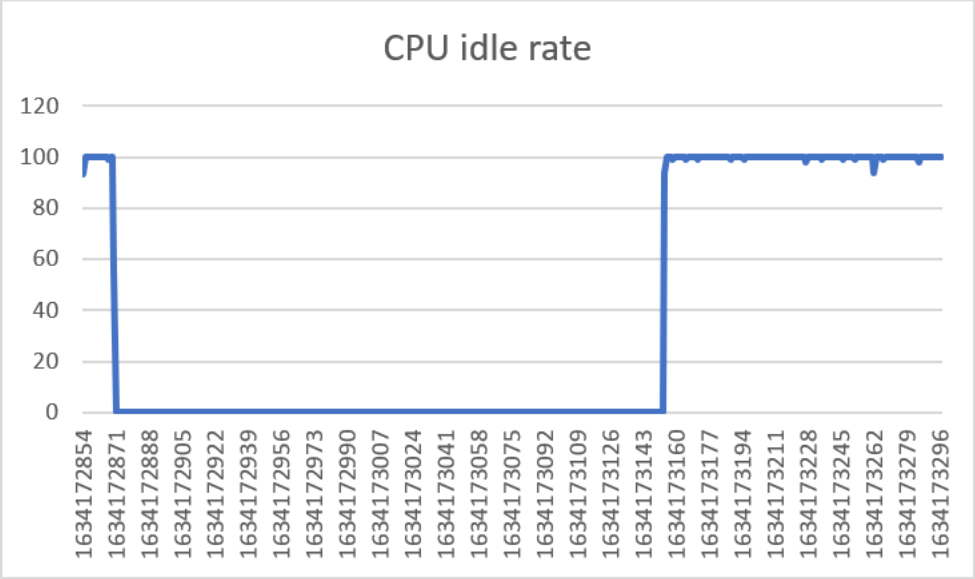

Figure 14 CPU utilization rate during the experiment

The power spectral density can be derived from the Fast Fourier Transform (FFT) to obtain the frequency in the amount of information gain observed during each trial. In Figure 15 the FFT for the QA and for the benchmark SA. The FFT output unit for magnitude is dB with a sampling frequency of 20.

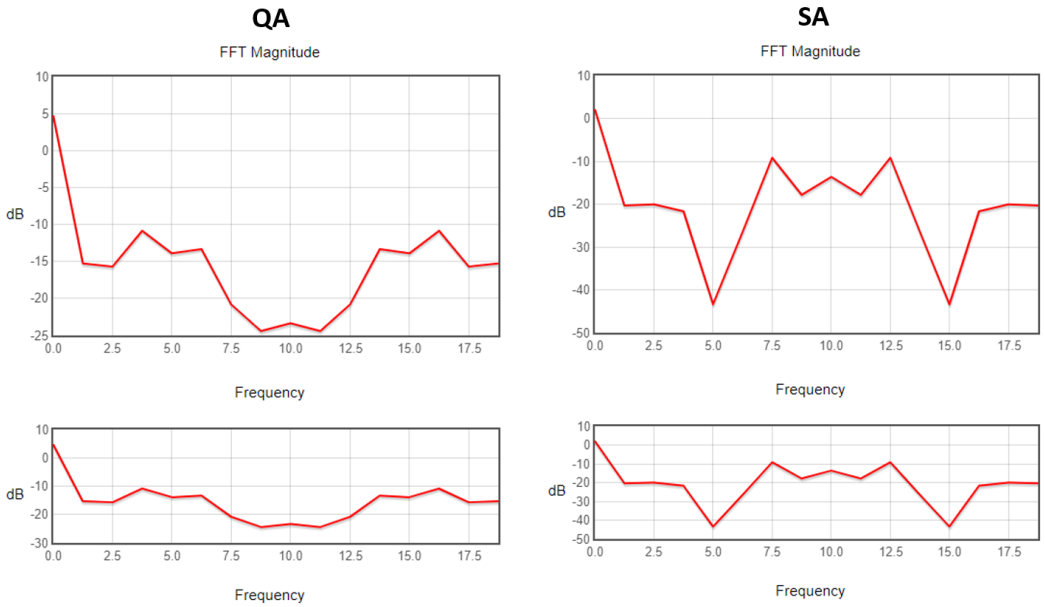

Figure 15 FFT of the rate of information gain during each trial for the SA and QA simulations.

We have performed a linear regression using the least square method to approximate the behavior of the algorithms to an equation. The output cost function relationship for QA and SA final cost is illustrated in Figure 16 and the details described in

Table 10.

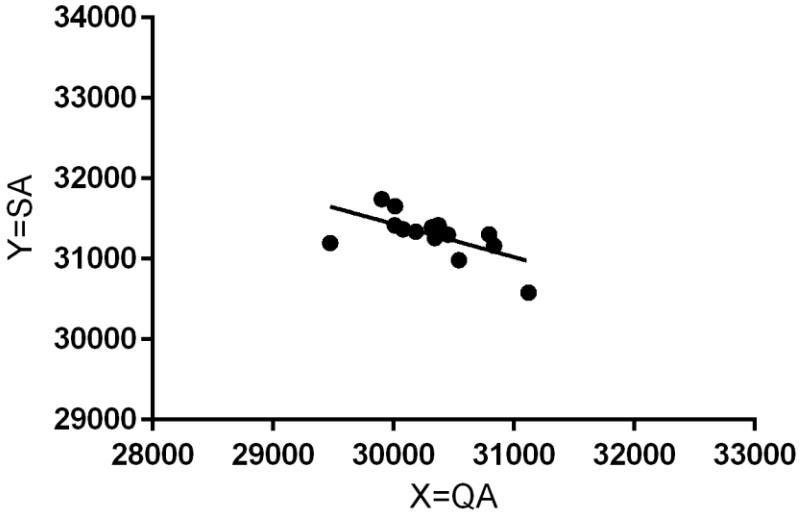

Figure 16 Linear regression for final state cost between QA and SA methods. Where each point is the optimal string with the smallest observed cost. X = QA and Y = SA

Best-fit values	
Slope	-0.4113 ± 0.1432
Y-intercept	43776 ± 4343
X-intercept	106423
1/Slope	-2.431
Equation	Y = -0.4113*X + 43776

a)

Slope significantly non-zero	

F	8.25E+00
DFn,DFd	1,13
P Value	0.0131
Significant	Yes

b)

Table 10 Best-fit values a) and p-value b) for the linear regression of string cost for the QA and SA methods

The relationship between computing time and energy change is illustrated in Figure 17. The equation and details are demonstrated in

Best-fit values	
Slope	35.12 ± 0.02345
Y-intercept	-0.01182 ± 0.01716
X-intercept	0.0003365
1/Slope	0.02848
Equation	Y = 35.12*X - 0.01182

a)

Slope significantly non-zero	
F	2.24E+06
DFn,DFd	1,13
P Value	< 0.0001
Significant	Yes

b)

Table 11 where X is the computational time and Y is the expected energy in Joule per bytes.

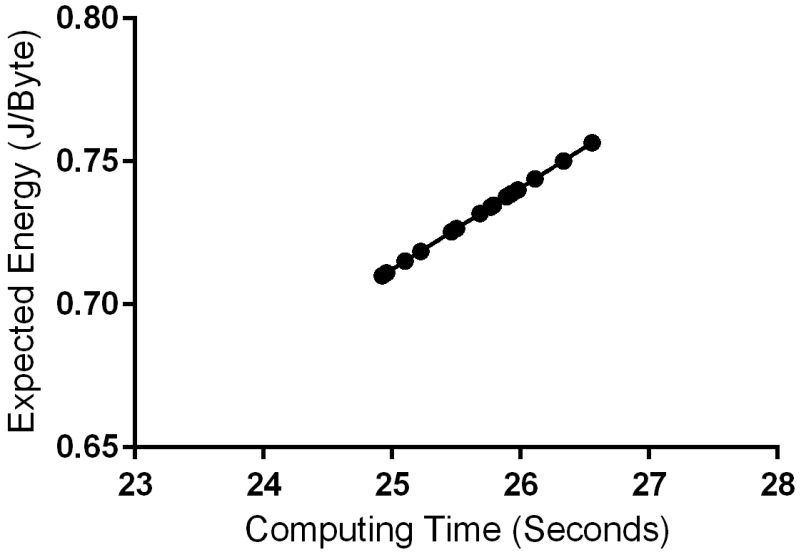

Figure 17 Linear regression for the relationship between total computing time in seconds (X) used by the machine and the associated energy requirement efficiency in Joules per bytes (Y).

Best-fit values	
Slope	35.12 ± 0.02345
Y-intercept	-0.01182 ± 0.01716
X-intercept	0.0003365
1/Slope	0.02848
Equation	Y = 35.12*X - 0.01182

a)

Slope significantly non-zero	
F	2.24E+06
DFn,DFd	1,13
P Value	< 0.0001
Significant	Yes

b)

Table 11 Best-fit values a) and p-value b) for the linear regression of computing time and energy change for the QA method.

Discussion & Conclusions

Discussion. From the above results we can conclude the following observations.

With a p-value less than 0.05 for the measured variables we can conclude that QA has statistically significant better observations than SA with the same system conditions and configurations.

Heuristic function QA requires less energy change per second for the same problem and input parameter as SA while generating near-optimal strings with less computation time and better utility cost quality. In other words, finding results with improved information gain per second, per joule.

The optimal string will be formed eventually with probability 1 for a long-term strategy without searching and sorting (i.e., by indexing) the entire search-space.

Each symbol in a candidate string directly affects the final solution quality for a process. We could infer the significance of a symbol by disabling their bits when forming the alternative string set and compare the output information gain to the results computed with the default language schema, with all symbols enabled. Symbols that are close to each other could be merged under a significance level (i.e., the probability of rejecting the null hypothesis given that its true).

The proposed method QA has a better noise-to-work ratio and thus having a better efficiency in terms of information gain per machine-cycle than the benchmark.

The energy computational model of non-polynomial programs coded and computed by a Turing Machine forms a noisy-communication channel that is bounded by the heat and noise limits, in a thermodynamic system, for a given bandwidth as defined by the Shannon-Hartley theorem.

The amount of energy needed to compute a program can be measured in Joule per bits and this value depends on the machine energy efficiency and the associated asymptotic complexity of the encoded problem.

There is a minimum amount of energy change lost to noise that is independent of the Language and depends only on the energy efficiency of system. Each system is comprised of a network of machines processing a program.

These results are in the interest of many engineering problems such as the optimization of problem-solving in computational systems. In those cases, the amount of energy needed to represent and process a bit of information can have a direct impact in the implementation and in the design of the machine components. This paper provided a new approach to computational complexity models as it provides insights in energy requirements to describe and process information within a network of interconected machines.

From the Halting problem and Computational Complexity Theory we can't know with probability 1, in *every* iteration cycle, without computing all elements in the search space, simultaneously:

Constraint 1: What is the shortest program, with minimal code length, that outputs the optimal string?

In other words, the string with the minimal entropy state.

Constraint 2: What is the statistical significance of a random candidate to be the optimal solution string X?

that simultaneous:

a) maximizes the confidence level of an arbitrary utility function u(X)
b) u(X) is a log-normal output distribution of a coded problem
c) The machine has finite cycles and power capacity

However, we can infer the minimal energy change requirements in terms of Joule per bits needed by a program to move from the initial - high entropy - state to the final - low entropy - state. This is the rate of energy transfer required by a program changing bits from a storage region of higher to lower entropy.

The energy requirement and the loss to heat is additive and the program of any complexity will be bounded by this heat. This increases the error-rate in the

channel as electrons move between atoms due to change in voltage potential in the circuit and causes increase in interference such as with thermal and quantum noise. This quantity of energy converted to heat and noise is proportional to the search space size, the utility function coded in the program, and the maximum frequency rate of bits computed by the machine. A real-world implementation of such machine will also dissipate heat in the network wires as they have an internal intrinsic resistance.

Conclusion. We can't know the exact amount of energy needed that optimizes the work-noise ratio for a program-machine system. Broadly speaking, the objective of this process is to move from state function A (initial, high entropy) to B (final, low entropy), with the least heat and noise possible, the smallest program length and the optimal string produced in the output. The meaning of an *optimal solution* string in this paper is interpreted as the method with:

1. Shortest code-length program.
2. The maximization of the dependent utility function of the described problem and the respective input string.
3. Computed with the minimum energy lost to heat or noise.

The exact energy of the system is almost impossible to measure. The non-computability of the Halting Problem suggests that if we knew the exact energy to communicate and compute, we could design a system that would have an energy efficiency coefficient closer to 1, with probability 1. If this holds almost all energy could be converted to work of string entropy reduction operations, with minimal conversion rate to heat and noise.

As demonstrated in this paper, we can however infer the minimum quantity of energy change needed to successfully describe and compute a bit of information, under a degree of entropy variance.

This information can be used to optimize process scheduling and network routing in distributed computing systems that have energy constraints and are bounded by a finite source of power.

References

Benjegerdes, T. (2009). *A Thermodynamics approach to High Performance Computing*. Retrieved from Sandia CA Seminar: http://bitspjoule.org

Björnson, E., & Larsson, E. (2018). How Energy-Efficient Can a Wireless Communication System Become? *Asilomar Conference on Signals, Systems, and Computers*, 1252-1256.

Hylton, T., Conte, T., & Hill, M. (2021). A Vision to Compute like Nature: Thermodynamically. *Communications of the ACM*, 35-38.

Lima, M. (2021). Information theory inspired optimization algorithm for efficient service orchestration in distributed systems. *PLOS ONE*.

Sun, H., Zhou, F., & Hu, R. (2019). Joint Offloading and Computation Energy Efficiency Maximization in a Mobile Edge Computing System. *EEE Transactions on Vehicular Technology, vol. 68, no. 3*, 3052-3056.