Dynamics of Algorithmic Processing in Computer Systems

Dominik Strzałka[*]
Franciszek Grabowski[†]
Department of Distributed Systems
Rzeszów University of Technology
W. Pola 2, 35-959 Rzeszów, Poland
[*]*strzalka@prz.edu.pl*
[†]*fgrab@prz.edu.pl*

This paper presents a new analytical and experimental approach to the insertion sort algorithm and task processing dynamics. The dependencies that exist in the task structure can influence the algorithm's behavior, especially in the number of dominant operations that are needed to compute the computational complexity. The proposed approach is based on a Tsallis definition of entropy suitable for all systems that are far from thermodynamical equilibrium. The ideas presented show how a complex systems approach can provide a good perspective for analyzing the processing dynamics of computer systems that are no longer simple.

1. Introduction

In order to understand the behavior of algorithms and the resources necessary for their computation, we need some measure. According to Rabin's proposal [1] and Turing's definition of an algorithmic machine [2], the degree of difficulty for some sets of functions is based on the number of steps necessary to achieve a solution (usually called time complexity) or the amount of tape necessary for computation (usually called memory complexity). The term "computational complexity" was proposed in 1965 by Hartmanis and Stearns [3].

Classical computational complexity makes two important assumptions. One is to minimize the influence of specific instances of input sets. Usually only the worst-case time complexity is considered. This is because the worst case is an upper bound for the observable running time. The second assumes that the time complexity should be independent of the specific central processing unit (CPU) clock rate, thus the total number of dominant operations needed for the computation is taken. However, this leads to another tacit assumption that the cost of all operations in an algorithm (or more generally, in a computer system) is exactly the same. For simplicity, we use big O notation because the exact number of steps will depend on the CPU or language being used for the implementation. If an instance of a problem

https://doi.org/10.25088/ComplexSystems.19.1.73

that is n bits long can be solved in n^3 steps, the problem has a time complexity of $O(n^3)$. If a problem has a time complexity of $O(n^2)$ on one CPU, then it will have that same complexity on most other CPUs, so this notation allows us to ignore the details of a particular computer.

Computational complexity describes how the running time and memory requirements increase as the size of the input grows. The theory places practical limits on what computers can accomplish. Many authors claim that computational complexity gives a full, detailed description of the algorithm's time action [4], and gives a description about changes in running time and memory requirements. As it turns out, these statements seem to be true only in special cases—the worst and the best—because in these cases the running time changes in a deterministic way. But what should be done if the complexity for the worst and best cases varies? Usually the computational complexity for the mean case is taken, as introduced in 1984 by Levin [5] to describe such instances for **NP** problems that can be solved in **P**. For **P** algorithms, the average-case computational complexity shows that every algorithm with polynomial time has polynomial time on average [6] and as a result the description by O notation is similar to the worst case. So this approach is not suitable for describing the dynamical behavior of any algorithm.

This paper consists of five sections. Section 2 describes insertion sorting and the possible dynamical behavior of computer algorithmic processing with a comparison to fluid flow. Section 3 shows a thermodynamical approach to analyzing computational complexity. Empirical results of our investigations are presented in Section 4.

2. Insertion Sort Algorithm

Knuth states that sorting is still one of the most important problems in computer processing, taking about 60 to 70% of total processing time [7]. From a theoretical viewpoint, the sorting problem is algorithmically closed: solutions exist that have a worst-case complexity equal to the lower limit for the sorting problem (i.e., $O(n \log n)$). But due to the importance of this problem and the new view proposed in this paper, we are still interested in it.

We consider the insertion sort algorithm, which is one of the simplest sorting procedures based on the behavior of a bridge player sorting their cards before the hand is played (see Figure 1).

The algorithm has two loops. The external loop guarantees that all sorting elements (keys) will be sorted, and an internal loop finds the right place for each sorted key. The computational complexity, based on the number of internal loop executions, is $O(n^2)$ in the worst case, but is $\Omega(n)$ for the best case. In these two cases the number of dominant operations needed for each sorted element can be determined.

But from a dynamical viewpoint they are not interesting because they are deterministic, that is, the number of dominant operations can be described by a mathematical equation [4]. Between these two cases exist $n! - 2$ other cases and such an equation cannot be written for many of them. They can lead to the dynamical behavior of the analyzed algorithm.

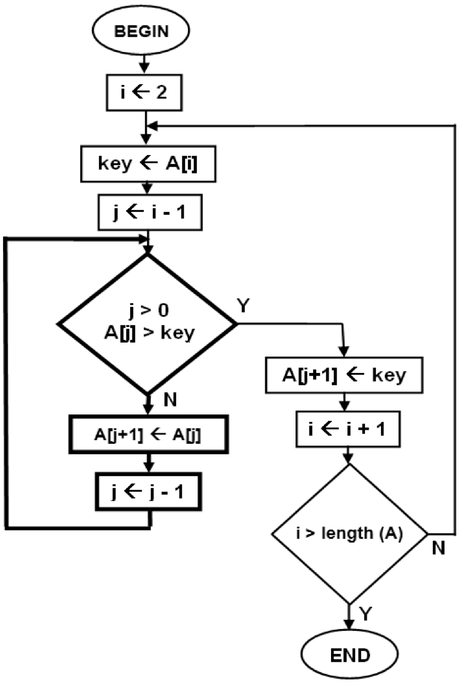

Figure 1. Flow chart for the insertion sort algorithm.

Note that in classical computational complexity the intrinsic features of *task* (i.e., the input data structure) are not taken into account. This is a natural consequence of the assumptions in this theory but one should realize that a data structure is not purely deterministic or purely random. For example, in the space of graphs, those that are regular or random can be analyzed, but some graphs have the small-world property. The small-world property does not make sense for small n [8]. The same situation occurs when there are (long-range) dependencies between values of successive keys. Such properties appear only when there is an appropriately long set (i.e., an appropriate task size). Another important example can be the space of elementary cellular automata. There are the completely regular automata behaviors, and there are those that are essentially random, but those that make spontaneous structure like rule 110 do not lend themselves to asymptotic analysis [9]. Other examples of such a behavior can be found in

[10]. The first works about computational complexity unjustifiably assumed that the analyzed algorithm would always behave in the same manner, no matter how large the input set. But today these assumptions are not sufficient and we need to go a step beyond.

What is meant by the dynamical behavior of an algorithm? In the insertion sort algorithm, finding the correct place for each sorted key is achieved by executing the internal loop. This loop is a trap for each sorted element. The problem is how long this loop will live. The optimistic and pessimistic cases are easy, because in the first case there are no executions and in the second case, for each successive element, the loop will live one dominant operation longer. As can be seen in Figure 2, the increments of this process are constant: 0 for the optimistic case and 1 for the pessimistic case.

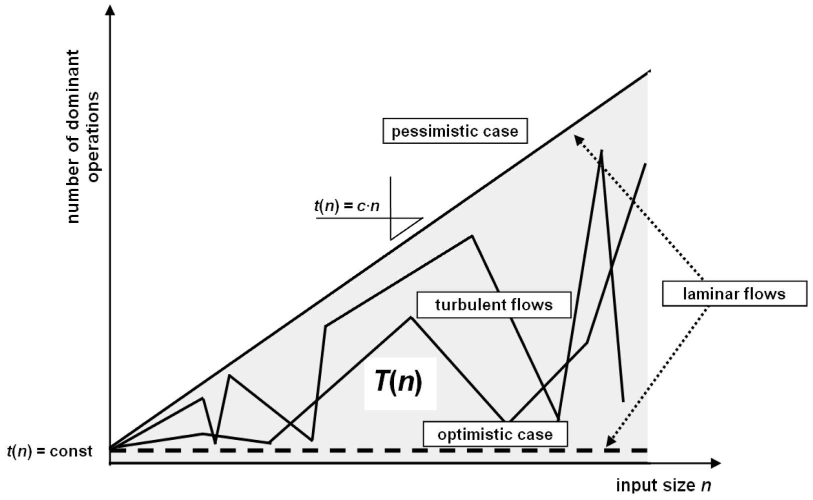

Figure 2. A possible dynamic behavior of the sorting algorithm. The optimistic and pessimistic cases are similar to laminar flow, while the other cases can have a turbulent nature [11].

What would happen if other cases were to appear? What is the incremental behavior of the number of dominant operations? If we compare sorting (or processing) to fluid flow, the pessimistic and optimistic cases are similar to deterministic laminar flow, while others are rather similar to nondeterministic turbulent flow (Figure 2).

Figure 3 shows a record of the dominant operations necessary for sorting one input set. For each sorted key in the pessimistic case, the number of dominant operations increases by 1, thus the slope is $\alpha_1 = 1$. But, due to some other (nonpessimistic and nonoptimistic) properties of the input set, it can also happen that a case has a slope of $\alpha_2 \gg \alpha_1$. From a dynamical viewpoint there can even exist cases when $\alpha_3 < 0$ or $\alpha \to \infty$. This is possible due to the strong dependence on input data features in the behavior of the insertion sort algorithm.

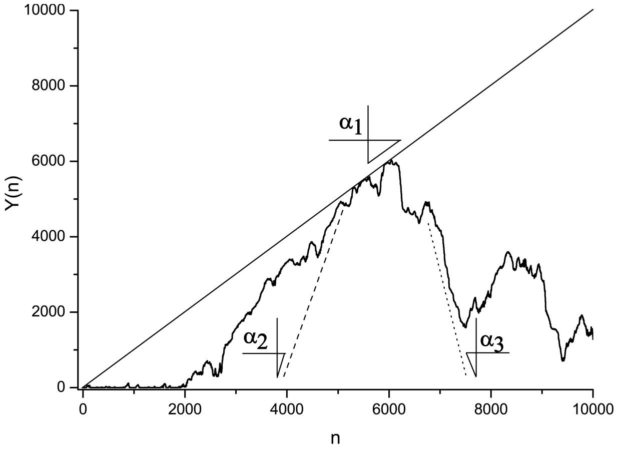

Figure 3. In many cases the incremental behavior in the number of dominant operations can be higher than for the worst case. From a dynamical viewpoint, such a situation is worse than in the pessimistic case.

A dynamic behavior can be identified with properties of the process that show how many dominant operations are needed for each sorted element. If the input data contains a decreasing trend, the number of dominant operations will rise and vice versa. The total number of dominant operations used in the algorithm depends on two conditions. The first is the index value of the sorted key; a larger index value guarantees that the loop lives longer. The second is the value of the sorted key; a higher value means a shorter loop duration. These two conditions come directly from the algorithm's flow chart realization (Figure 1). However, there is also a third condition hidden inside the algorithm: the duration of the internal loop also depends on the elements that have been sorted so far. This means that in the insertion sort algorithm a feedback loop exists that also has an influence on its dynamics. Due to this feedback a parasite path appears (see Figure 5). The parasite path later represents loops with varying lifetimes and shows why there can exist cases with a slope of $\alpha_2 \gg 1$ or even a slope of α_3 with a negative sign (Figure 3).

If sorting processes can have dynamical properties that are similar to nondeterministic turbulent flow, can the processes be described by the classical definition of the Boltzmann-Gibbs (BG) entropy? If someone sorts, they also order, so in other words the entropy level of the input set decreases, but of course the entropy of the environment increases. We can show a new approach that goes further than allowing assumptions in the classical computational complexity analysis and rejecting the minimization of task dependency on the complexity measure, where one takes into account only a behavior of total time in the worst case. The existing simple system approach is based on the assumption that computer processing is strictly algorithmic. But computer processing can also be interactive [12] and the existing approach limits any considerations about the possible dynamic behavior of many algorithms. This behavior can have a detailed description only by using a complex systems approach.

3. Thermodynamical Approach to Computational Complexity

When will the entropy flow in the analyzed algorithm be at its lowest possible level [11]? Let us assume that n is the size of a sorted set, n_i is the number of successively sorted elements, and M stands for the total number of internal and external loop executions; thus, $M = n_i$ for each sorted key. Further, let M_1 denote the number of external loop executions. In the analyzed algorithm for each sorted key M_1 is always equal to 1. M_2 will similarly denote the number of the internal loop executions and it can change from 0 to $n_i - 1$. Finally, M_3 will denote the number of possible internal loop executions that can be done, but do not appear due to some properties of the input set. Obviously, $M = M_1 + M_2 + M_3$. Now we can define a number W of possible configurations of loop executions that can appear during sorting the best, the worst, and other cases. This allows us to consider thermodynamic properties of the sorting process for each sorted set of size n. The number W of possible combinations is equal to $C_M^{M_1}$ multiplied by $C_{M_1}^{M_2}$:

$$W = C_M^{M_1} \cdot C_{M_1}^{M_2} = \frac{M!}{M_1!\,(M - M_1)!} \cdot \frac{(M - M_1)!}{M_2!\,(M - M_1 - M_2)!} = \frac{M!}{M_1!\,M_2!\,M_3!}. \tag{1}$$

For the optimistic case we have one execution of the external loop ($M_1 = 1$), no executions of the internal loop ($M_2 = 0$), and $M_3 = n_i - 1$ possible internal loop executions that will not be executed. In the optimistic case W_O equals

$$W_O = \frac{n_i!}{1!\,0!\,(n_i - 1)!} = n_i. \tag{2}$$

For the pessimistic case the situation is similar: $M_1 = 1$ gives only one execution of the external loop, the internal loop used $M_2 = n_i - 1$

executions, and obviously $M_3 = 0$. Thus, the number W_P in the pessimistic case equals

$$W_P = \frac{n_i!}{1!\left(n_i - 1\right)!\,0!} = n_i. \tag{3}$$

The results obtained are a little surprising (we found that $W_O = W_P$) but it should be noted that the pessimistic case means an inverse order in the input set, although that is still some kind of order. In these two cases we have the lowest possible level of entropy production and from a thermodynamical viewpoint they are almost indistinguishable. In any other case W will be greater. For example, consider the case when the key n_i needs only one excess dominant operation, that is, $M_1 = 1$, $M_2 = 1$, $M_3 = n_i - 2$, and the number of microconfigurations W_D will equal

$$\begin{aligned} W_D &= \frac{n_i!}{1!\,1!\left(n_i - 2\right)!} = \\ &\frac{\left(n_i - 2\right)!\left(n_i - 1\right)!\,n_i}{\left(n_i - 2\right)!} = n_i\left(n_i - 1\right) \end{aligned} \tag{4}$$

and $W_D > W_O$.

By using Onsanger relations [13], the lowest possible level of entropy production indicates that the system is in a state close to (thermal) equilibrium. For such a situation the BG entropy is proper. But, equation (1) shows the level of entropy production for each successively sorted key and immediately another question arises: is this process extensive (additive) for all n_i elements? In the classical approach for computational complexity analysis one of two similar ways can be followed. In the first a total time (i.e., a total number of dominant operations) $T(n)$ can be considered, while in the second the number of dominant operations $t(n)$ (i.e., increments of total time $T(n)$) for each element of the input set can be analyzed. The second approach is more interesting because it will give an answer to the stated question. We do not need knowledge about the distribution behavior of W for each key, but (in the simplest approach) information on the distribution behavior of all possible increments of the number of dominant operations, that is, $t'(n) = t(n+1) - t(n)$. Thus, for each sorted key n_i the value of $t'(n)$ is a random variable s_i, with values that appear with changing probabilities. We would like to know the distribution of the random variable $S = \sum_i s_i$ when $n_i \to n$ and of course when $n \to \infty$.

Some properties of S can be deduced without visualization. For example, successive s_i will always have a finite value of variance, thus S should also have this property. This suggests that the probability distribution of S for $n_i \to n$ will be in the Gaussian domain of attraction. But how will the distribution behave for smaller values of n_i? To answer this question, we calculate such a distribution for a set consisting of $n = 32\,000$ elements (Figure 4).

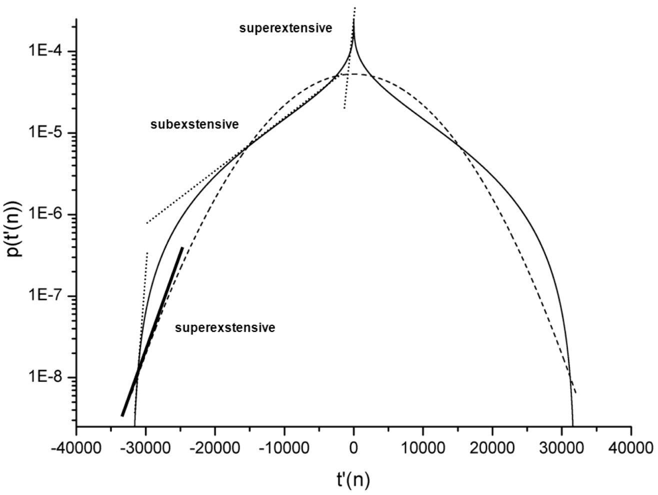

Figure 4. Distribution of random variable S shows changes in the value of the q parameter.

In [14] Tsallis showed that the different values of q in his definition of entropy lead to different symmetric probability distributions. Even with heavy tails for $q > 5/3$, Figure 4 shows that the distribution of random variable S has parts where the tail vanishes at different rates. This suggests that in some cases there can appear a probability distribution of increments $t'(n)$ that has a thermodynamical basis that can be described only by the Tsallis nonextensive entropy. When can such cases appear? This depends on properties of the input sets and is why we propose retiring the existing assumption about the independence of input data properties and the time computational complexity. The existing assumption does not allow showing a real, possible complexity of algorithmic behavior. The independence between data structure (input task) and algorithm behavior is proper for the simple systems approach used in computer systems analysis. But the computer systems are no longer simple systems due to their specific properties; they are now complex.

4. Some Properties of Sorting Dynamics

To visualize the considerations presented, we take into account different sets of input data and show that the dynamic behavior of the insertion sort algorithm depends on the input data. Investigations were based on 1000 arbitrarily chosen trajectories of fractional Brownian motion with $n = 10^6$ keys. Such trajectories can have a long-term memory property (if $H > 0.5$) that appears by the existence of increasing and decreasing trends (we used 500 input sets with $H = 0.5$ and 500 sets with $H = 0.9$). Such trends are the local pessimistic and optimistic cases for the sorting algorithm and should influence the dynamics of the sorting process. Before the experiment, we assumed that the long-term memory property in the input data could be carried into the dynamics of the sorting process. To confirm our assumptions, we made a small modification in the original algorithm code and introduced the variable `counter` to count how many dominant operations were needed for each sorted key. As a result, the time series could be analyzed further.

The internal loop in the insertion sort algorithm plays a very important role from a dynamical viewpoint. The number of internal loop executions is deterministic in the optimistic and pessimistic cases. When comparing the sorting process to fluid flow, each sorted key would show a laminar flow. To help understand this, we now show how the successively sorted keys travel through the flow chart of the insertion sort algorithm shown in Figure 1. First, we make a few modifications in the original flow chart to show what is going on with each processed key (Figure 5). In the optimistic case the first condition (namely, $A[i] > \max$) is checked; if it is true, then a key follows a path marked by the solid line. If all keys follow this path we have the optimistic case. In the pessimistic case a condition $A[j] < \min$ is checked; if it is true, then a key follows a path marked by the dotted line. If all keys follow this path we have the pessimistic case. But for other cases, successive keys follow a third path (similar to the second path, but different from a dynamical viewpoint); a situation that has appeared leads to the existence of a parasite path in the algorithm marked by the dashed line.

Because we used the comparison of the internal loop to a trap, we analyzed the probability distribution of trap lifetimes. Kernel estimators were used that allowed us to get a continuous probability distribution, which was then plotted on a log-log scale. As a result of this, we saw that the power law governs trap lifetime processes (Figure 6). The power law distribution has appeared for almost all sorting cases, independently of the value of the H exponent in the input data.

In other words, trap lifetimes have a fractal property that does not have a characteristic scale and can lead to excess $1/f$ noise. Such a distribution cannot have its thermodynamical basis described by BG extensive entropy.

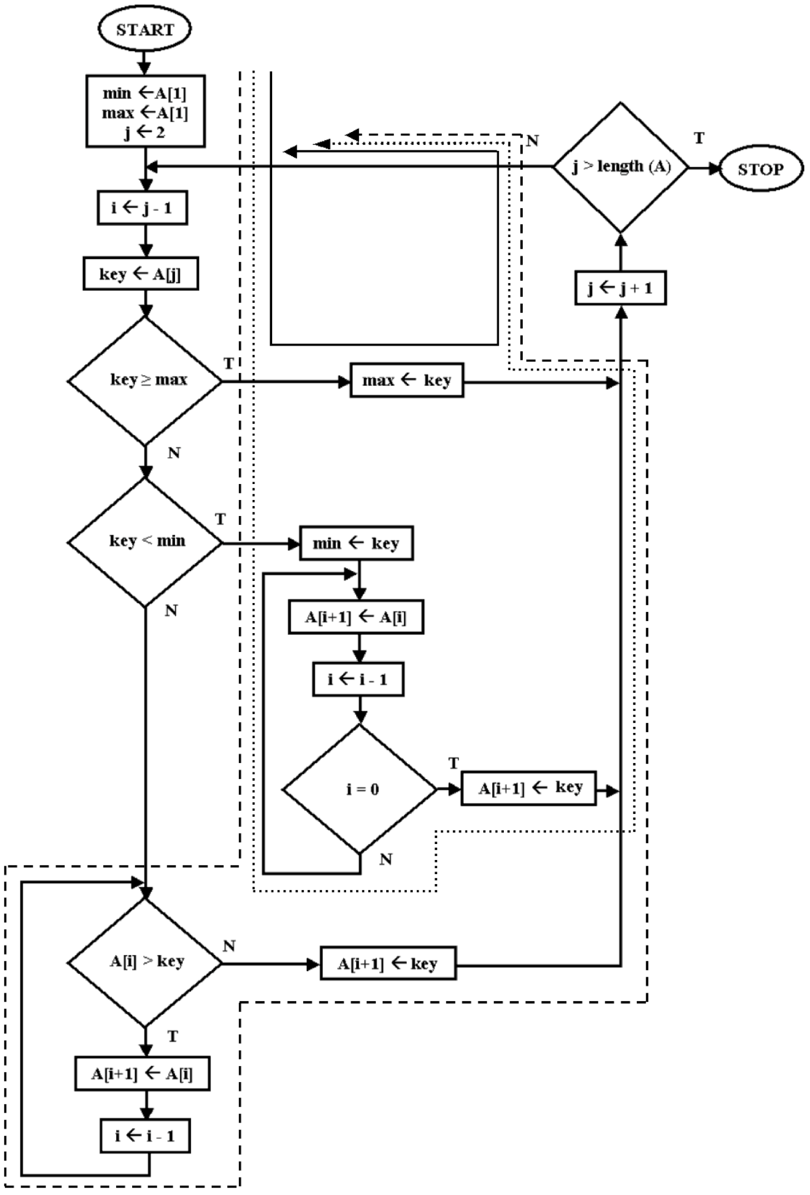

Figure 5. During sorting the keys follow three different paths. The first (solid line) is proper for successive elements with max value, the second (dotted line) is for elements with min value, and the third (dashed line) for the rest shows the existence of a parasite loop in the insertion sort algorithm.

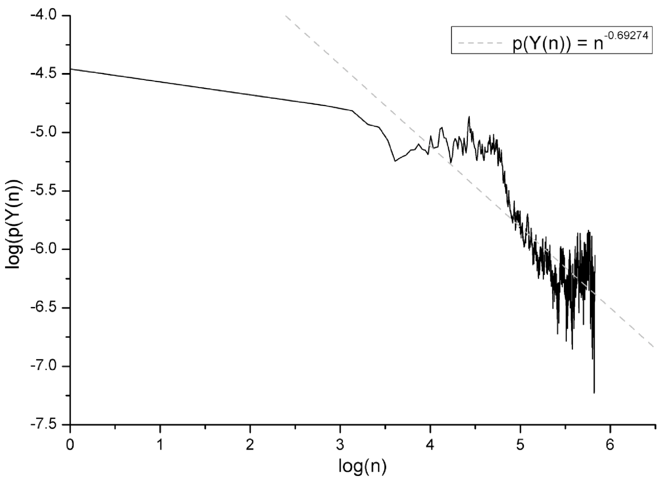

Figure 6. The power law that governs the distribution of trap lifetimes.

Then we took into account the increments of time $t(n)$ (i.e., $t'(n)$) and as a result got a realization of an incremental process similar to that of noise. This operation allowed us to make further investigations; quantile lines tests decided whether processes are stationary or not and analysis of probability distributions led to the assumption that the time series are asymptotically stationary. Their probability distribution was especially interesting. Plotting the values in a log-lin scale showed that the incremental process has properties that can only appear by the nonextensive definition of entropy because their tails vanish slowly when compared to fitted Gaussian distributions (Figure 7). This result once again confirms that the Tsallis entropy is proper for the thermodynamical basis of our analysis.

One interesting property we used for sorting the different sets of input data is that half of them have long-range dependencies. Immediately one question arose: does the task with the long-term memory property influence the sorting dynamics? To check this we used Hurst *R/S* analysis and detrended fluctuation analysis [15], both of which confirmed our expectations: if long-range dependencies existed in the input data, they influenced the behavior of the insertion sort algorithm. The dynamics of the sorting process depend on this property and can also have it (Figure 8).

The existence of thermodynamical nonequilibrium states during sorting should lead to the appearance of bifurcations [13]. Such bifurcations exist and can be visualized with special diagrams (Figure 9) where the number of dominant operations necessary for each sorted key versus key values ordered inversely are plotted. Each decreasing trend in the input set implies the existence of a local bifurcation.

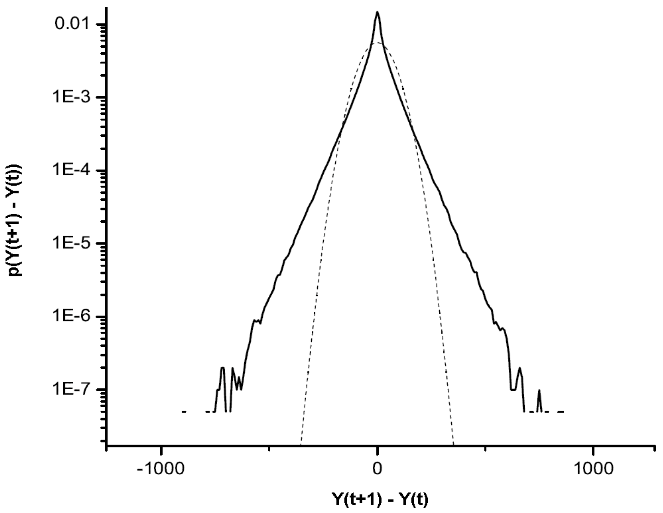

Figure 7. Semi-heavy tails (solid line) in the empirical distribution of increments of dominant operations for successive sorted keys compared with a normal distribution (dashed line).

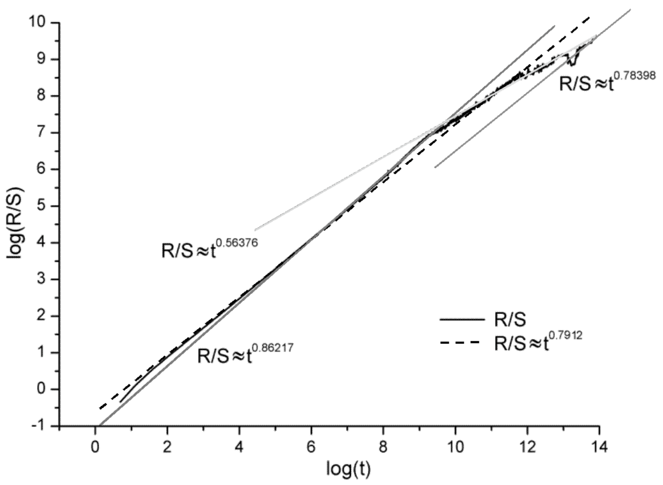

Figure 8. *R/S* analysis confirms the existence of long-range dependencies in the sorting dynamics when such a property exists in the input data.

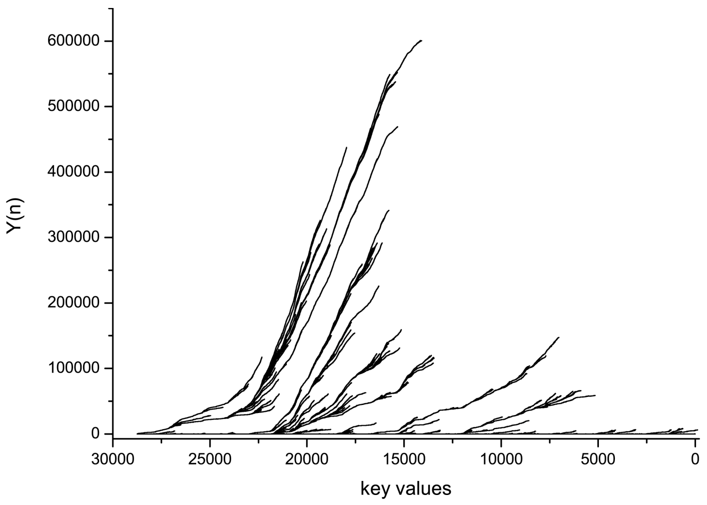

Figure 9. Bifurcation diagram for the sorting process.

The thermodynamical nonequilibrium states found in computer processing led us to another interesting observation: referring back to Figure 3, it can be seen that in the pessimistic and optimistic cases the number of dominant operations are simply a straight line that has a dimension $D = 1$. Mandelbrot, in the famous article [16] states: “Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line.” We now can ask: does the computational complexity of an algorithm always follow a straight line or can this line have fractal properties sometimes? It can be seen that this line can have a dimension $D = 2 - H$, that is, $D > 1$ (see Figure 8). As a result of this finding the authors would like to introduce in a future work a new (possible) measure of algorithm complexity, that is, a fractal computational complexity. Such a measure would describe a real complexity of algorithm behavior according to fractal properties of the input data. The current approach describes rather a “simplicity” (not a “complexity”) of the computational complexity due to considering only the worst and best cases (also including the mean-case analysis).

5. Conclusions

We have shown that classical computational complexity analysis, although it gives a very useful basis for algorithm analysis, cannot describe the dynamics in the behavior of the insertion sort algorithm. Such a property governed the realization of the sorting process, due to the existence of long-range dependencies in the input sets. This case shows that laws that were previously unknown govern algorithmic processing in computer systems. Only the complex systems approach, which goes beyond the classical computational complexity assumptions and takes into account the behavior of an algorithm together with the properties of the input data, allows a better understanding and gives a good perspective for the future analysis of the complex behavior of computer systems that is far from thermodynamical equilibrium.

References

[1] M. O. Rabin, "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets," *Tech. Rep. No. 2*, 1960, Jerusalem, Israel: Hebrew University.

[2] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, Series 2, **42**, 1936 pp. 230-265. Errata appeared in Series 2, **43**, 1937 pp. 544-546.

[3] J. Hartmanis and R. E. Stearns, "On the Computational Complexity of Algorithms," *Transactions of American Mathematical Society*, **117**(5), 1965 pp. 285-306.

[4] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Cambridge, MA: MIT Press, 1994.

[5] L. A. Levin, "Problems Complete in 'Average' Instance," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC 1984)*, Washington, D.C., New York: ACM Press, 1984.

[6] J. Wang, "Average-Case Computational Complexity Theory," in *Complexity Theory Retrospective II* (L. Hemaspaandra and A. Selman, eds.), New York: Springer, 1997 pp. 295-328.

[7] D. Knuth, *The Art of Computer Programming I*, Reading, MA: Addison-Wesley, 1997.

[8] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small World' Networks," *Nature*, **393**, 1988 pp. 440-442.

[9] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[10] J. Smith, "RandScape: Complex Images from Simple Algorithms," *Artificial Life*, **9**(1), 2003 pp. 67-78.

[11] D. Strzałka and F. Grabowski, "Towards Possible Non-Extensive Thermodynamics of Algorithmic Processing—Statistical Mechanics of Insertion Sort Algorithm," *International Journal of Modern Physics C*, **19**(9), 2008 pp. 1443-1458.

[12] P. Wegner and D. Goldin, "Computation beyond Turing Machines," *Communications of the ACM*, **46**(4), 2003 pp. 100-102.

[13] I. Prigogine and I. Stengers, *Order out of Chaos: Man's New Dialogue with Nature*, New York: Bantam Books, 1984.

[14] C. Tsallis, S. V. F. Levy, A. M. C. Souza, and R. Mayanard, “Statistical-Mechanical Foundation of the Ubiquity of Lévy Distributions in Nature,” *Physical Review Letters*, **75**(20), 1995 pp. 3589-3593.

[15] C-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger, “On the Mosaic Organization of DNA Sequences,” *Phys. Rev. E*, **49**, 1994 pp. 1685-1689.

[16] B. B. Mandelbrot, “How Long Is the Coastline of Great Britain, Statistical Self Similarity and Fractional Dimension,” *Science*, **155**, 1967 pp. 636-638.