Stable Sorting Using Special-Purpose Physical Devices

Niall Murphy[1], Damien Woods[2], and Thomas J. Naughton[1]

[1] Dept. of Computer Science, National University of Ireland, Maynooth, Ireland
nmurphy@cs.nuim.ie, tom.naughton@nuim.ie
[2] Boole Centre for Research in Informatics, School of Mathematics, University College Cork, Ireland
d.woods@bcri.ucc.ie

BCRI Preprint 06/2006
May 2006
Boole Centre for Research in Informatics, University College Cork, Ireland

Abstract. We define a computational model of physical devices that have a parallel atomic operation that transforms the input in such a way that the sorted list can be sequentially read off in linear time. We show that several commonly-used scientific laboratory techniques (from biology, chemistry, and optical physics) are instances of the model. We show how our model naturally suggests an improvement in functionality (specifically, stability) to an existing physics-inspired sort that is not stable.

1 Introduction

There has been interest in identifying, analysing, and utilising computations performed in nature [1, 3, 7–9, 11, 14, 17, 18, 20], in particular where they appear to offer reduced complexity solutions when compared with the best-known sequential (e.g. Turing machine) equivalent. In this paper we present a special-purpose unconventional model of computation that falls into that category.

The special-purpose model works as follows. A one-dimensional (1D) input vector is transformed to a two-dimensional (2D) matrix via a constant-time atomic operation. This 2D matrix representation admits a simple linear time algorithm that produces a (stable) sort of the original 1D input list.

This model has at least three implementations utilised frequently by scientists in the fields of chemistry, biology and optical physics. In these capacities it is used to sort particles of diameter in the order of 10^{-6} meters and below. This suggests that the model can be used for massively parallel operations. The three implementations presented in this paper are routinely used for ordering physical objects but to our knowledge, have never before been proposed for sorting lists of numbers. Other natural sorting algorithms have been proposed in the literature [2, 3, 15].

In Section 2 we introduce this Model of Physical Sorting (which we simply refer to as the Model). Then in Section 2.1 we provide a high level description of the Model before providing a more formal definition in Section 2.2. One of the interesting aspects of the Model is that several implementations of it already exist; in Section 3 we describe three implementations that are used as real-world sorting methods. In Section 4 we show how this model compares to an existing physically inspired sort called Rainbow Sort [15]. On the one hand we give a generalisation of Rainbow Sort in Section 4.1 and show that it is an instance of our model. On the other hand we restrict our model in Section 4.2 so that it accurately captures the standard Rainbow Sort. Section 5 concludes the paper.

2 Model of Physical Sorting

In this section we introduce the Model of Physical Sorting. It is inspired by the observation that a physical force affects particles differently depending on certain physical properties of those particles. We use this effect for sorting by encoding an unordered list of numbers as a list of particles with appropriate properties.

2.1 Informal description

We define a Model of Physical Sorting whose instances take as input a list $L = (l_1, l_2, \ldots, l_n)$ and compute the *stable sorting* [10] of the list. A sorting is stable if and only if sorted elements with the same value retain their original order. More precisely, a stable sorting is a permutation $(p(1), p(2), \ldots, p(n))$ of the indices $\{1, 2, \ldots, n\}$ that puts the list elements in non-decreasing order, such that $l_{p(1)} \leqslant l_{p(2)} \leqslant \cdots \leqslant l_{p(n)}$ and that $p(i) < p(j)$ whenever $l_{p(i)} = l_{p(j)}$ and $i < j$. (Not all sorting algorithms are stable; we give some counterexamples. Clearly any sorting algorithm that does not preserve the original relative ordering of equal values in the input is not stable. A sorting algorithm that relies on each element of its input being distinct to ensure stability is not stable. A sorting algorithm that outputs only an ordered list of the input elements (rather than indices) cannot gaurantee stability.) The Model produces a stable sorting as follows. The input list is transformed to a 2D matrix that has a number of rows equal to the input list length and a number of columns linear in the maximum allowable input value. The matrix is zero everywhere except where it is populated by the elements of the input list, whose row position in the matrix is their index in the input and whose column position is proportional to their value. The values in the matrix are then read sequentially, column by column, and the row index of each nonzero value is appended to an output list. This output list of indices is a stable sorting of the input list.

2.2 Formal description

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. We begin by defining our sorting model.

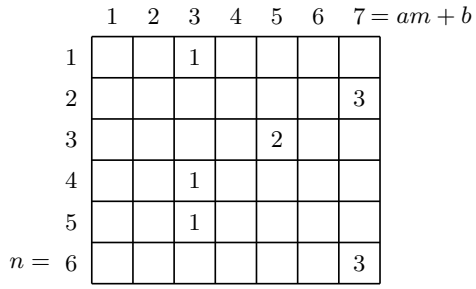

Fig. 1. Graphical illustration of the matrix G for example model $S = (m, a, b) = (3, 2, 1)$ and for example input list $L = (1, 3, 2, 1, 1, 3)$.

Definition 1 (Model of Physical Sorting). *A Model of Physical Sorting is a triple $S = (m, a, b) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, where m is an upper bound on the values to be sorted and a, b are scaling constants.*

The Model acts on a list $L = (l_1, l_2, \ldots, l_n)$ where $l_i \in \{1, \ldots, m\}$. Given such a list L and a Model of Physical Sorting S we define a $n \times (am + b)$ matrix G with elements

$$G_{i,j} = \begin{cases} l_i & \text{if } j = al_i + b \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

An example G for given S and L is shown in Fig. 1.

Definition 2 (Physical Sorting computation). *A Physical Sorting computation is a function $c : \{1, \ldots, m\}^n \to \{1, \ldots, n\}^n$ that maps a list L of values to a sorted list*

$$c(l_1, l_2, \ldots, l_n) = (k_1, k_2, \ldots, k_n) \tag{2}$$

of indices, where l_{k_p} is the p^{th} non-zero element of G and the elements of G are assumed to be ordered first by column and then by row.

Remark 1. A Physical Sorting computation returns a stable sorting of its input: k_1 is the index of the first value in the stable sorting of L, k_2 is the index of the second value, and so on.

Remark 2. We assume that a Physical Sorting computation is computed in exactly $(am + b)n + 1 = O(n)$ timesteps. The creation of matrix G takes one timestep and obtaining the indices of the nonzero values in G takes one timestep per element of G.

Each of the physical instances of our Model of Physical Sorting that follow are consistent with Remarks 1 and 2; the matrix G is generated in a single parallel

timestep and the Physical Sorting computation takes linear time to output a stable sorting.

An interesting feature of the algorithm is the fact that it has a parallel part followed a sequential part. One could ask that the entire algorithm be either entirely sequential or entirely parallel, with a respective increase or decrease in time complexity. However, neither of these scenarios correspond to the way in which the below physical instances are actually performed in the laboratory.

3 Physical Instances of the Model

In this section we give example instances of the Model that arise in commonly-used scientific laboratory techniques. The common idea behind these implementations is that some physical force will affect items to a greater or lesser extent depending on their physical properties.

We show that each of these examples implements the model's computation. We do this by specifying relevant values for the triple S (Definition 1) and by showing that the examples compute in a way that is consistent with Definition 2. Thus each example generates G (Equation (1)) and is a stable sort.

3.1 Gel Sort

Gel electrophoresis [16] is a fundamental tool of molecular biologists and is a standard technique for separating large molecules (such as DNA, RNA and polypeptide chains) by length. It utilises the differential movement of molecules of different sizes in a substance (an agarose gel) of a given density.

The process of gel electrophoresis occurs as follows. Each sample of uniform length molecules is placed at one end of a rectangle of agarose gel, separated from each other by being placed in wells at different spatial locations in a line. The gel is then permeated with a conducting liquid. Electrodes apply a voltage across the gel which provides a force upon the (sometimes artificially) charged molecules causing them to be pulled towards the oppositely charged electrode. Smaller molecules move through the gel more quickly and easily than larger molecules. This difference in velocity separates the molecules and orders them by length.

Viney and Fenton [19] provide an equation that describes the physics of gel electrophoresis,

$$V = K_1 \frac{E}{\varepsilon M^n} - K_2 E \,, \tag{3}$$

where V is the velocity of a molecule of molecular mass M in an electric field E, where the ratio between the pore size and the typical size of the molecules is given by $0 < n \leqslant 1$, and where ε is the permittivity of the conducting liquid. The constants K_1 and K_2 are composed of such variables as the length of the gel, and the charge per unit length of the molecule [19].

To get distance s we apply $V = s/t$ where t is time, giving

$$s = K_1 \frac{Et}{\varepsilon M^n} - K_2 Et \,.$$

We refer to sorting using gel electrophoresis as Gel Sort. For an instance of Gel Sort we choose appropriate values for $K_1, K_2, E, t, \varepsilon \in \mathbb{R}$ such that $k_1 = (K_1 Et/\varepsilon) \in \mathbb{N}$ and $k_2 = (K_2 Et) \in \mathbb{N}$. We also let $n = 1$ which gives

$$s = k_1 M^{-1} - k_2 \,. \tag{4}$$

Equation (4) satisfies Equation (1) if we let $S = (m, k_1, k_2)$ where $m \in \mathbb{N}$ is the largest length of DNA, RNA or polypeptide chain to be sorted.

Given a list L to be sorted, we encode each value as a molecule with length inversely proportional to each value. Each molecule is then placed in an individual well, in the same order of the list to be sorted L. After the gel is run, it is a representation of the matrix G. To read the list of indices corresponding to a stable sort, we sequentially record the indices of the samples beginning with those that travelled least. Thus Gel Sort implements arbitrary computations of the Model.

3.2 Optomechanical Sort

It is known that transparent objects experience a force when a beam of light passes through them [4]. This force is caused by the beam's path being refracted by the object. A change in light beam direction causes a change in the beam's momentum, and momentum is only conserved if there is an equal but opposite change of momentum for the object. This momentum change has a component in the same direction as the direction of the beam and a component in the direction of the increasing gradient of the beam (the gradient force, F_{grad}). This effect is most commonly employed in optical tweezers [5].

We use this technology to sort objects and we refer to this sort as Optomechanical Sort. In Optomechanical Sort, all of the input objects are arranged in a straight line in a medium (e.g. water). There is a barrier that prevents the objects from moving in the direction of the beam (the scattering force). A light source with a strictly increasing intensity gradient perpendicular to the axis of the input objects is supplied. The objects with a larger volume will move more quickly in the direction of increasing intensity than those of a smaller volume. This movement will separate the objects according to their volumes.

We will now proceed using the equations for objects smaller than the wavelength of the light beam. Ashkin [6] provides the equation to calculate the force in the direction of the gradient on the particles

$$F_{\text{grad}} = -\frac{n_b^3 V}{2}\left(\frac{m^2-1}{m^2-2}\right)\nabla E^2$$

here n_b is the refractive index of the medium, m is the refractive index of the particles divided by the index of the medium, V is the volume of the particles and ∇E^2 is the change in beam density over the particle.

For each instance of Optomechanical Sort we let $n_b, m, \nabla E^2$ be constants such that

$$F_{\text{grad}} = k_1 V \tag{5}$$

where $k_i \in \mathbb{N}$, holds. Equation (5) satisfies Equation (1) with $S = (m, k_1, 0)$ where m is the maximum particle volume for the specific material and medium. The sort is stable as we obtain a list of indices by reading the index of each particle in the order of least distance traveled and since the particles move in parallel lines. Thus Optomechanical Sort is an instance of the Model.

3.3 Chromatography

Chromatography is a collection of many different procedures in analytical chemistry [12] which behave similarly (e.g. gas chromatography, liquid chromatography, ion exchange chromatography, affinity chromatography, thin layer chromatography).

The result of chromatography is the separation of the sample input chemicals (analytes) over time. Chromatography achieves this by exploiting the behaviours of different chemicals in two media; the mobile phase and the stationary phase. The mobile phase is a solvent for the analytes and filters through the stationary phase. The stationary phase resists the movement of the analytes to different degrees based on their chemical propeties. This causes the analytes to separate over time.

We use standard equations from analytical chemistry [13] to calculate the distance traveled by an analyte in a particular mobile phase and stationary phase.

Given the time t_m for the mobile phase to travel distance L_m, the average velocity $\bar{u}_m$ of the mobile phase in the stationary phase and the capacity factor k of the analyte, Poole and Schuette [13] provide

$$t_R = \frac{L_m}{\bar{u}_m}(1 + k)$$

to find the time t_R that it takes the analyte to travel the distance L_m. They also provide

$$k = \frac{t_R - t_m}{t_m}$$

to find the value of k. By substitution we find

$$L_m = \bar{u}_m t_m \,,$$

It follows that an analyte moving at an average velocity of $\bar{u}_a \leqslant \bar{u}_m$ will in time t_m travel a proportional distance $L_a \leqslant L_m$, that is

$$L_a = \bar{u}_a t_m \,. \tag{6}$$

We refer to the use of chromatography to sort substances by their average velocity $\bar{u}_a$ through the stationary phase as Chromatography Sort. If we provide an instance of the Model with the triple $S = (m, t_m, 0)$ where $m = \bar{u}_m$ is the average velocity of the mobile phase in the stationary phase, it is clear that Equation (6) satisfies Equation (1).

Also, we ensure that Chromatography sort is stable by running each analyte to be sorted side by side, or on a separate but identical, apparatus. The final indices are read off in order of the distance traveled by the analyte. Thus Chromatography Sort is an instance of the model.

4 Rainbow Sort

Rainbow Sort was first described by Schultes [15]. It utilises the phenomenon of dispersion, where light beams of longer wavelengths are refracted to a lesser degree than beams of a shorter wavelength.

A set of elements is to be sorted. Each element is encoded as a beam of light of a distinct wavelength. The separate beams are then combined into one beam of light and this is passed through a prism. The component beams are refracted at different angles and so emerge from the prism separately and in an order dictated by their wavelength. A light measurement device can be positioned to sequentially read the ordered component beams.

There is a relationship between the angle of deviation δ (the angle between the input beam and the output beam) and the refractive index of the prism medium for each wavelength of light [15]. We will restrict the possible wavelengths of the input beam so that the distance from where the uninterrupted beam would have reached the sensor to where the diffracted beam reaches it is linear in $\tan\delta$ and the distance between the sensor and the prism. This can be expressed as

$$s = p \tan \delta \tag{7}$$

where s is the distance along the sensor and p is the distance between the sensor and the prism surface. Equation (7) satisfies Equation (1) if we let $S = (m, p, 0)$ where m is the minimum wavelength that can be diffracted by the implementation. However, Rainbow Sort cannot guarantee to return a stable sorting as it returns a sorted list of wavelengths not a list of indices. Also Rainbow Sort cannot be directly used to sort lists with repeated values. Rainbow Sort is not an instance of the Model.

4.1 Generalising Rainbow Sort

We now outline a natural generalisation of Rainbow Sort that satisfies our Model. We refer to this generalisation as Generalised Rainbow Sort. Generalised Rainbow Sort is very similar to Rainbow Sort except that it utilises the full geometry of the prism. It keeps each input beam at a different depth in the prism, as shown in Figure 2(B). The output is then detected by a two dimensional sensor and is read off sequentially in a manner consistent with Definition 2.

Generalised Rainbow Sort sorts the input in a manner that is similar to Rainbow Sort but instead returns a list of indices, and naturally deals with repeated elements in the input. The resulting Generalised Rainbow Sort is a stable sort. Using Equation (7) and the Model triple $S = (m, p, 0)$ from Section 4

we see that Generalised Rainbow Sort is an instance of the Model. It is interesting to note that the introduction of stability to Rainbow Sort does not decrease its big-Oh time complexity.

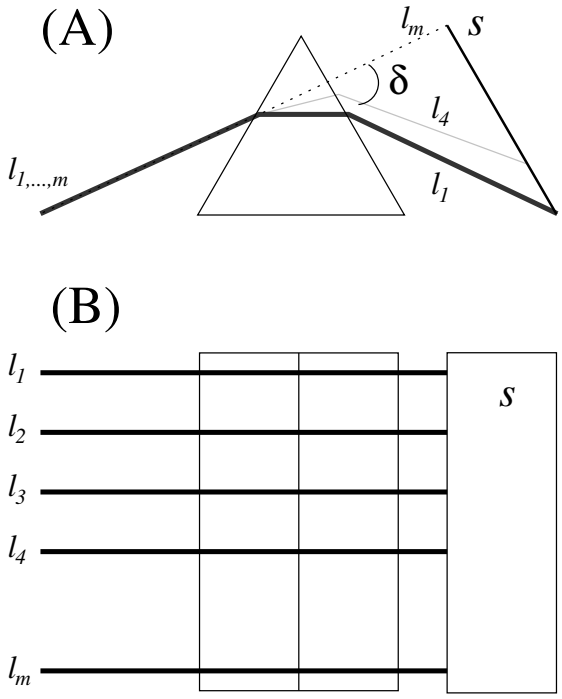

Fig. 2. Rainbow Sort and Generalised Rainbow Sort. (A) The computation of Rainbow Sort and also a side elevation of Generalised Rainbow Sort. Here s represents a sensor. The angle δ is the angle of deviation. (B) Top down view of Generalised Rainbow Sort.

4.2 Restricting the Model

We now restrict the Model to simulate the original Rainbow Sort [15]. To do this we redefine Equation (1) and restrict Definition 2 to one dimension. Definition 1 remains unchanged for the Restricted Model of Physical Sorting (the Restricted Model).

The Restricted Model model acts on a set $T = \{t_1, t_2, \ldots, t_n\} \subset \mathbb{N}$. Given such a set T and a Restricted Model of Physical Sorting S we define the vector V where $|V| = am + b$. As before a, b are scaling constants and $m = \max(T)$. The vector V has elements

$$V_j = \begin{cases} t_i & \text{if } j = at_i + b \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Definition 3 (Restricted Physical Sorting computation). *A Restricted Physical Sorting computation is a function c that maps a set* $T = \{t_1, t_2, \ldots, t_n\} \subset \mathbb{N}$ *to a list*

$$c(T) = (t_{k_1}, t_{k_2}, \ldots, t_{k_n}) \tag{9}$$

where t_{k_p} *is the* p^{th} *non-zero element of* V*.*

It is not difficult to see that $c(T)$ is strictly increasing, that is $t_{k_i} < t_{k_{i+1}}$ for all $i \in \{1, 2, \ldots, n-1\}$.

The Restricted Model computes a non-stable sorting of the input set. Equation (7) satisfies Equation (8) if we let the Restricted Model triple be $S = (m, p, 0)$ where m is the minimum wavelength that is diffracted by the implementation and p is as in Equation (7). Thus Rainbow Sort is an instance of the Restricted Model.

The physical instances of the Model in Section 3 can be suitably restricted to become instances of the Restricted Model and thus have equivalent sorting abilities to Rainbow Sort. This restriction is achieved by removing the abilities to track indices and deal with repeated elements.

To restrict Gel Sort all samples are placed in a single well. The output is an ordering of the samples by size. In order for the sort to be stable it must be case that all inputs are distinct. Thus this resticted Gel Sort is not stable. In Optomechanical Sort we direct the input set along an angle slanting across the intensity gradient so that the particles do not move in parallel but are forced into a single ordered line. We cannot distinguish identical input elements from the output. In Chromatography Sort we mix the analytes and separate them on the same column, the output is an ordered list however we can no longer identify separate identical input values. In order for the latter two sorts to be stable it must be case that all inputs are distinct, thus they are not stable.

5 Conclusion

In this paper we have proposed a Model of Physical Sorting that computes a stable sorting of its input list of natural numbers. This model has a parallel 1D to 2D (list to matrix) transformation as an atomic operation, where only one dimension of the matrix is dependent on the input list length. Once in matrix form it becomes a linear-time sequential task to read the list of stable sorted indices. As examples of physical sorts that are instances of the Model, we have provided three physical implementations that are well-known laboratory techniques from experimental science. Other well-known example candidates are centrifugal separation and fractional distillation [12].

We showed how the Model naturally suggests how to introduce stability into an existing physics-inspired sort, Rainbow Sort. Finally, we showed that a restricted version of the Model accurately describes (the standard) Rainbow Sort, and further show that the three physical instances of the Model described in this paper (from biology, chemistry, and optical physics) have similar restricted implementations with the same properties and time complexity as (the standard) Rainbow Sort.

References

1. Leonard Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021 – 1024, 1994.
2. Dewdney A.K. On the spaghetti computer and other analog gadgets for problem solving. *Scientific American*, 250(6):19 – 26, jun 1984.
3. Joshua J. Arulanandham, Cristian S. Calude, and Michael J. Dinneen. A fast natural algorithm for searching. *Theoretical Computer Science*, 320:3 – 13, 2004.
4. Arthur Ashkin. Acceleration and trapping of particles by radiation pressure. *Physical Review Letters*, 24(4):156 – 159, 1970.
5. Arthur Ashkin. History of optical trapping and manipulation of small-neutral particle, atoms, and molecules. *IEEE Journal on Selected Topics in Quantum Electronics*, 6(6), 2000.
6. Arthur Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and Steven Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics Letters*, 11(5):288 – 290, 1986.
7. Yaakov Benenson, Tamar Paz-Elizur, Rivka Adar, Ehud Keinan, Zvi Livneh, and Ehud Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414:430 –434, 2001.
8. Cristian S. Calude and Gheorghe Păun. *Computing with Cells and Atoms*. Taylor & Francis Publishers, London, 2001.
9. Thomas Head. Formal language theory and dna: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 47(6):737 – 759, 1987.
10. Donald E. Knuth. *The Art of Computing Programming: Sorting and Searching*, volume 3. Addison-Wesley, second edition, 1997.
11. Carver Mead. *Analog VLSI and Neural Systems*, chapter 6, pages 83–99. Addison-Wesley, Reading, Massachusetts, 1989.
12. Clifton E. Meloan. *Chemical Separations: principles, techniques, and experiments*. Wiley-Interscience, 1999.
13. Colin F. Poole and Sheila A. Schuette. *Contemporary practice of chromatography*. Elsevier, 1984.
14. Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108 – 143, 2000.
15. Dominik Schultes. Rainbow sort: Sorting at the speed of light. *Natural Computing*, 5(1):67 – 82, 2006.
16. Colin F. Simpson and Mary Whittaker, editors. *Electrophoretic techniques*. Academic Press, London, 1983.
17. Tommaso Toffoli. What are nature's 'natural' ways of computing? In *PhysComp '92 – Proceedings of the Workshop on Physics of Computation*, pages 5 – 9, 1992.
18. Tommaso Toffoli. Programmable matter methods. *Future Generation Computer Systems*, 16:187 – 201, 1999.
19. Christopher Viney and Richard A. Fenton. Physics and gel electrophoresis: using terminal velocity to characterize molecular weight. *European Journal of Physics*, 19(6):575–580, 1998.
20. Damien Woods and Thomas J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1 – 3):227 – 258, apr 2005.