Implementations of a model of physical sorting

Niall Murphy[1], Thomas J. Naughton[1], Damien Woods[2], Beverley Henley[3], Kieran McDermott[3], Elaine Duffy[4], Peter J. M. van der Burgt[4], and Niamh Woods[4]

[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
[3] Dept. of Anatomy, BioSciences Institute, University College Cork, Ireland
b.henley@ucc.ie
[4] Dept. Experimental Physics, National University of Ireland, Maynooth, Ireland
peter.vanderburgt@nuim.ie

Abstract. We define a computational model of physical devices that have a parallel atomic operation that transforms their input, an unordered list, in such a way that their output, the sorted list, can be sequentially read off in linear time. We show that several commonly-used scientific laboratory techniques (from biology, chemistry, and physics) are instances of the model and we provide experimental implementations.

1 Introduction

There has been interest in identifying, analysing, and utilising computations performed in nature [1, 7, 8, 10, 11, 16, 19, 21, 23, 26, 28], in particular where they appear to offer interesting resource trade-offs when compared with the best-known sequential (e.g. Turing machine) equivalent. In this paper we present a special-purpose model of computation that falls into that category. Other natural sorting algorithms have been proposed in the literature [2, 3, 24].

Natural scientists routinely separate millions of particles based on their physical characteristics. For example, biologists separate different lengths of DNA using gel electrophoresis [25], chemists separate chemicals by using chromatography [22], and physicists separate particles based on their mass-to-charge ratio using mass spectrometry [15]. The common idea behind these techniques is that some physical force affects objects to an amount that is proportional to some physical property of the objects. We use this idea to sort objects and provide a special-purpose model of computation that describes the method idea formally. In this paper we present five instances of the model that are routinely used for ordering physical objects but, to our knowledge, all but one have never before been proposed for sorting lists of numbers. We also provide four physical implementations utilised frequently by scientists in the fields of chemistry, biology, and physics. In these fields, particles of diameter 10^{-6} meters and below are sorted. This suggests that the model can be used for massively parallel computations.

The special-purpose model works as follows. We encode the list of numbers (input vector) in terms of one physical property of a chosen class of particle, such that an easily realisable known physical force will affect proportionally each particle based on its value for that property. The one-dimensional (1D) input vector is transformed to a two-dimensional (2D) matrix via a constant-time atomic operation which represents the action of the force. This 2D matrix representation admits a simple linear time algorithm that produces a (stable) sort of the original 1D input list.

In Section 2 we introduce the Model of Physical Sorting (which we simply refer to as the Model). We define the (general) Model in Section 2.1 and the Restricted Model 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 five instances and present experimental results for four that are used as real-world sorting methods. Some of these instances are not instances of the Model but of the Restricted Model. In Section 3.4 we show how a pre-existing instance of the Restricted Model is generalised to become an instance of this Model. Section 4 concludes the paper.

2 Model of Physical Sorting

In this section we introduce the Model of Physical Sorting and the Restricted Model of Physical Sorting. Their instances take as input a list $L = (l_1, l_2, \ldots, l_n)$ and compute the *stable sorting* [17] of the list.

Definition 1 (Stable sort). *A sort 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 inputs being distinct to ensure stability is not stable. A sorting algorithm that outputs only an ordered list of the input elements (rather than indices) is not necessarily stable.

2.1 The Model

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. Before formally describing the computation of the Model we give an informal description. 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

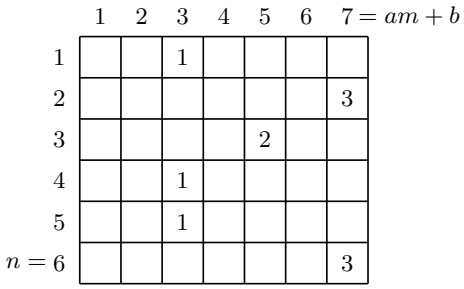

Fig. 1. Graphical illustration of the matrix G for example model $S = (m, a, b) = (3, 2, 1)$ and for example input list $L = (1, 3, 2, 1, 1, 3)$.

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.

Definition 2 (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 $l_i \in \{1, \ldots, m\}$ and m is some constant that is independent of n. 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 Figure 1.

Definition 3 (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 of indices*

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

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 at most $(am + b)n + 1 = O(n)$ 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 physical instances (see Section 3) are actually performed in the laboratory.

2.2 Restricted Model

It is possible to restrict some physical instances of the Model in Section 3 to become instances of the Restricted Model. This restriction is achieved by removing the abilities to track indices and deal with repeated elements.

Definition 4 (Restricted Model of Physical Sorting). *A Restricted 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 Restricted Model acts on a multiset $T = \{t_1, t_2, \ldots, t_n\}$, $t_i \in \mathbb{N}$ and m is some constant that is independent of n. Given such a multiset T and a Restricted Model of Physical Sorting S we define the vector V of length $am + b$. As before a, b are scaling constants and $m = \max(T)$. The vector V has elements

$$V_j = \begin{cases} t_i & \text{if } j = at_i + b \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Definition 5 (Restricted Physical Sorting computation). *A Restricted Physical Sorting computation is a function c that maps a set $T = \{t_1, t_2, \ldots, t_n\}, t_i \in \mathbb{N}$ to a list*

$$c(T) = (t_{k_1}, t_{k_2}, \ldots, t_{k_n}) \tag{4}$$

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

It is not difficult to see that $c(T)$ is a list of strictly increasing values, 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 multiset.

Remark 3. The input to a Restricted Model of Physical Sorting is a multiset, however the output vector does not contain any duplicated elements. Also the output of a Restricted Physical Sorting computation is a sorted list of the input elements, no index information is available and so is not necessarily a stable sort.

Remark 4. We assume that a Restricted Physical Sorting computation is computed in at most $(am + b) + 1 = O(1)$ timesteps. The creation of vector V takes one timestep and obtaining the sorted list in V takes one timestep per element of V.

3 Physical Instances of the Model

In this section we give five example instances of either the Model, the Restricted Model that arise in commonly-used scientific laboratory techniques, gel electrophoresis, chromatography, the dispersion of light, optical tweezers, and mass spectrometry. A brief introduction is given to each instance and we explain how it is used to sort. Most of the examples are instances of both the Model and the Restricted Model and we show how the examples compute in a way that is consistent with the models. In all but one case (optical tweezers) we present an experimental implementation of the physical instance.

With Gel Sort and Rainbow Sort the matrix produced is the mirror image of the other examples; larger input elements have a smaller column index in the matrix while smaller input elements have a large column index. Reading the matrix of sorted values in the normal way yields an output list of non-increasing order. To get an output list of non-decreasing order we read the matrix starting with the largest column index.

3.1 Gel Sort

Gel electrophoresis [25] is a fundamental tool of molecular biologists and is a standard technique for separating large molecules (such as DNA and RNA) by length. It utilises the differential movement of molecules of different sizes in a gel of a given density.

Description. The process of gel electrophoresis is illustrated in Figure 2 and occurs as follows. Samples of DNA molecules are placed into wells at one end of a rectangle of agarose gel. The wells are separated from each other at different spatial locations along a straight line. The gel is then permeated with a conducting liquid. Electrodes apply a voltage across the gel which provides a force upon the 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 orders the molecular samples by number of base pairs.

Sorting. Here we briefly describe Gel Sort; using gel electrophoresis for sorting. Given a list L of numbers to be sorted, we encode each element of L as a sample of molecules with a number of base pairs proportional to the element value. Each sample of uniform length molecules is placed (in the same order as in L) in the wells at one end of the gel. A voltage is applied for a time and the molecules move through the gel at a rate inversely proportional to their length. When the voltage is removed the gel is a representation of the matrix G (see Definition 2). We then read off the list of sorted indices by recording the index of each element in order of those which traveled the least and in order of their index. The resulting list is in decreasing order. Restricted Gel Sort is similar to Gel Sort except that all the samples of molecules are placed in the same well.

Instance. Viney and Fenton [27] provide an equation that describes the physics of gel electrophoresis,

$$V = K_1 \frac{E}{\varepsilon M^n} - K_2 E \,, \tag{5}$$

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 represent quantities such as the length of the gel, and the charge per unit length of the molecule [27].

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

$$s = k_1 M^{-1} - k_2 \,. \tag{6}$$

Equation (6) satisfies Equation (1) if we let $S = (m, k_1, k_2)$ where $m \in \mathbb{N}$ is the smallest length of DNA or RNA to be sorted.

Given a list L to be sorted, we encode each list value as a sample of molecules of proportional length. Each sample of molecules 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 traveled least. Thus Gel Sort implements arbitrary computations of the Model.

Given a multiset T to be sorted, we encode each multiset value as a sample of molecules of proportional length. If we place all molecules in one well, after the gel is run it is a representation of the vector V. We read the sorted list by recording the length of each sample beginning with those that traveled least. Thus Restricted Gel Sort is an instance of the Restricted Model of Physical Sorting.

Implementation We have implemented an instance of Gel Sort and Restricted Gel Sort. The elements of the unordered list of numbers to sort $L =$ (550, 162, 650, 200,550, 350, 323, 550) are encoded as DNA strings with a number of base pairs proportional to their value. These values of DNA are placed in individual wells in a 1% agarose gel in the order that they appeared in the list L. We also place a sample of each in a single well to produce a non-stable sort for comparison. The gel is run for some time and the result is seen in columns $1-8$ in Figure 2. We then read off the list of sorted indices by recording the index of each element in order beginning with those which traveled the least and in order of their index. This yields the list of indices $c(L)$ =(2,4,7,6,1,5,8,3) which yields the sorted list of elements (162, 200, 323, 350, 550, 550, 550, 650).

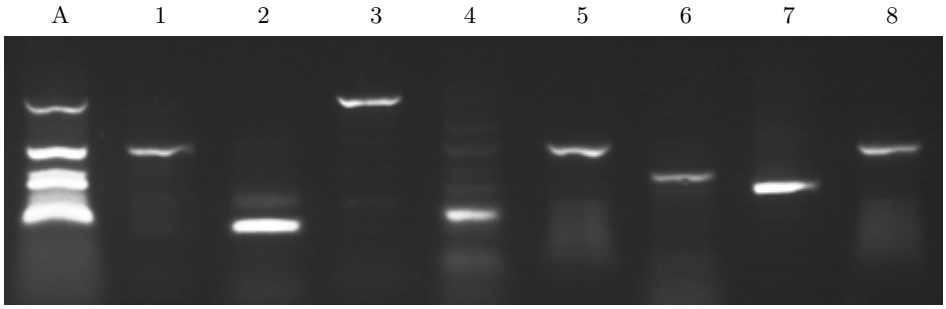

Fig. 2. Electrophoresis of DNA molecules in a 1% agarose gel. Lane A is a non-stable sort which contains strings of DNA with the same number of base pairs as all of those in lanes 1 to 8. In lanes 1 to 8 the DNA molecule lengths are respectively 550, 162, 650, 200, 550, 350, 323 and 550 base pairs.

We have also performed a Restricted Gel Sort that is an instance of the Restricted Model of Physical Sorting (see Definition 5). We sort the set of numbers $T = (550, 162, 650, 200, 550, 350, 323, 550)$ but this time they are all placed in a single well in the agarose gel. The result which is seen in column A of Figure 2 is $c(T) = (162, 200, 323, 350, 550, 650)$. This is a non-stable sorting of the list L as the multiple instances of the element 550 were lost in the result.

3.2 Optomechanical Sort

The movement of small transparent particles by light alone is an effect most commonly employed in optical tweezers [5] for biologists to manipulate micro-scale objects. Several methods of ordering particles using this technology have been proposed [9, 14]. We, however, propose a novel method that is an instance of the model of Physical Sorting. We do not provide a implementation of Optomechanical Sort or an instance of the Restricted Model.

Description 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 intensity gradient of the beam (the gradient force, F_{grad}).

Sorting. We propose the use of optical tweezers 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

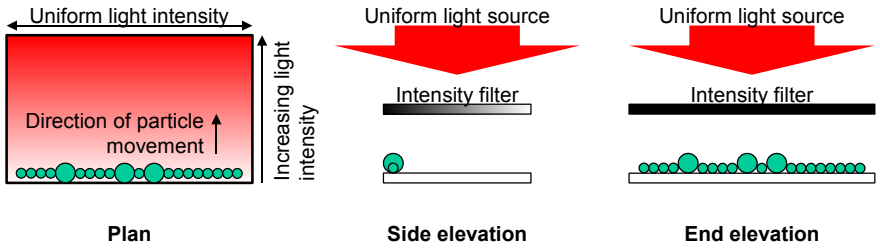

Fig. 3. The initial configuration of Optomechanical Sort. The circles represent the particles to be sorted.

beam. A light source, constant in time, and with a strictly increasing intensity gradient perpendicular to the axis of the input objects is applied (see Figure 3). This intensity gradient is achieved by modulating a uniform light field with an intensity filter variable in one direction only. The objects with a larger volume move more quickly in the direction of increasing intensity than those of a smaller volume. This movement separates the objects according to their volumes.

Instance We proceed by using the equations for objects smaller than the wavelength of the light beam. According to Ashkin [6] the equation to calculate the force in the direction of the gradient on the particles is

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

where 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

$$F_{\text{grad}} = k_1 V \tag{7}$$

where $k_1 \in \mathbb{N}$, holds. Equation (7) 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 Sort

Chromatography is a collection of many different procedures in analytical chemistry [20] which behave similarly (e.g. gas chromatography, liquid chromatography, ion exchange chromatography, affinity chromatography, thin layer chromatography). It is commonly used to separate the components in a mixture.

Description. Chromatography separates the input chemicals (analytes) over time 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 properties. This causes the analytes to separate over time.

Sorting. We refer to the use of chromatography to sort substances by their average velocity through the stationary phase as Chromatography Sort. We proceed assuming known relative velocities for analytes in our apparatus. The apparatus is either wide enough to accommodate many analytes side by side or is made of several identical setups which allow side by side comparisons.

Given a list L of numbers to be sorted, we encode each element of L as a sample of analyte with a relative velocity proportional to the element value. Each analyte is placed in the chromatography apparatus in the same order as in L. When the process commences the analytes move along the stationary medium at a rate proportional to its relative velocity. When the process is halted the apparatus is a representation of the matrix G (from Equation (1)). We then read off the list of sorted indices by recording the index of each element in order of those which traveled the most and in order of their index (position in L).

Restricted Chromatography Sort is similar to Chromatography Sort except that all analytes are mixed together and placed in the apparatus as one sample.

Instance. We use standard equations from analytical chemistry [22] 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 [22] 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

$$L_a = \bar{u}_a t_m \,. \tag{8}$$

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

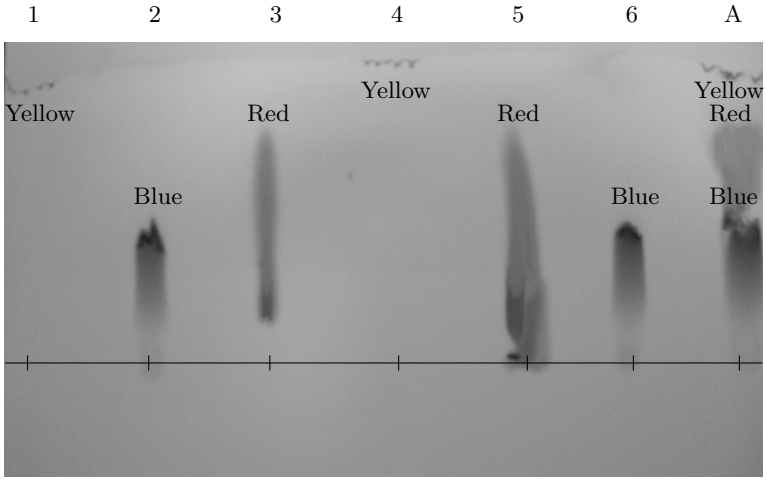

Fig. 4. Chromatography of household food dye in water on a thin layer plate. Lane A is a non-stable sort which contains each of the dyes in lanes 1 to 6. In lanes 1 to 6 the average speeds are, from left to right 0.00003ms^{-1}, 0.00001ms^{-1}, 0.00002ms^{-1}, 0.00003ms^{-1}, 0.00002ms^{-1}, 0.00001ms^{-1}.

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 (8) 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. After the run, the apparatus is a representation of the matrix G. The final indices are recorded in order of analytes that traveled the least distance and in the case of analytes traveling the same distance, in the order of the indices starting with the smallest. Thus Chromatography Sort is an instance of the model.

In an instance of Restricted Chromatography Sort all analytes are mixed together and placed in the apparatus as one sample. After the run the apparatus is a representation of the vector V. The sorted list is read by recording the order of the analytes staring with those that traveled the least distance. Restricted Chromatography Sort is an instance of the Restricted Model.

Implementation. We have performed an implementation of Chromatography Sort with a list of numbers $L = (3, 1, 2, 3, 2, 1)$. The chemicals involved were domestic food dyes and their average velocities were measured with a water mobile phase and thin layer stationary phase. We assigned each element in L a chemical directly proportional to its average velocity with a water mobile phase and thin layer stationary phase. In this case the assignment was 1 to blue, 2 to red, and 3 to yellow.

The result is seen in Figure 4. The final indices are read off in order of least distance traveled by the analytes giving the list $c(L) = (2,6,3,5,1,4)$. These

indices yield the stable sorting of L with duplicates in the same order that they appeared in the input. We also performed an instance of the Restricted Model, the output of which is seen in Figure 4 column A. The input set was $T = (3, 1, 2, 3, 2, 1)$ and the numbers were encoded using the same scheme. The sorted output is $c(T) = (0.00001\text{ms}^{-1}, 0.00002\text{ms}^{-1}, 0.00003\text{ms}^{-1})$ which does not record the multiple instances of set elements. The result is a non-stable sorting of the set T.

3.4 Rainbow Sort

Rainbow Sort was first described by Schultes [24] as an unstable sort but by a simple generalisation it becomes an instance of the Model.

Description. Rainbow Sort utilises the phenomenon of dispersion, where light beams of longer wavelengths are refracted to a lesser degree than beams of a shorter wavelength. Dispersion occurs where there is a change of refractive index in the media (such as an interface between air and glass) in the path that the light beam travels.

Sorting. In Rainbow Sort, as described by Schultes [24] each element of a list L is encoded as a distinct wavelength proportional to its value. A beam of light containing only the wavelengths to be sorted is passed through a prism. The component wavelengths are refracted at different angles and so emerge from the prism as separate beams and in an order dictated by their wavelength (see Figure 5). A light measurement device is positioned to sequentially read the ordered component beams. Schultes considers the input encoding (which takes linear time) in his complexity analysis which differs from our constant time analysis. This is an unstable sort as it does not output repeated elements that were in the input and as defined does not return a list of indices and so is not necessarily stable.

Schultes provides a possible technique to sort lists with repeated elements with Rainbow Sort [24]. We suggest our own method that follows from the Model of Physical Sorting called Generalised Rainbow Sort, which is similar to Rainbow Sort except that it utilises the full geometry of the prism and is an instance of the Model. It also returns the indices of the sorted list elements, thus guaranteeing stability. Each element of the list L is encoded as a beam of light of a distinct wavelength proportional to its value. Each beam is then passed through the prism at a different depth in the prism, as shown in Figure 5. We then read off the list of sorted indices by recording the index of each refracted beam in order of those which where refracted the most and were there are multiple beams refracted to the same degree, in order of their index.

Instance. 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 [24]. We limit the set of available

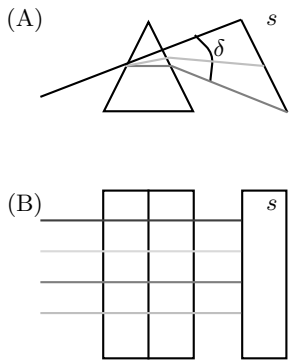

Fig. 5. Rainbow Sort and Generalised Rainbow Sort. (A) The computation of Rainbow Sort and also a side elevation of Generalised Rainbow Sort. Here s represents a sensor. The angle δ is the angle of deviation. (B) Top down view of Generalised Rainbow Sort.

input beam wavelengths so that the distance from where the uninterrupted beam would have reached the sensor to where the diffracted beam reaches it is linear in both $\tan\delta$ and the distance between the sensor and the prism. This is expressed as

$$s = p \tan \delta \tag{9}$$

where s is the distance along the sensor (see Figure 5) and p is the distance between the sensor and the prism surface. Equation (9) satisfies Equation (3) if we let the Restricted Model triple be $S = (m, p, 0)$ where m is the minimum wavelength whose refracted path can be measured by the implementation and p is as in Equation (9). Rainbow Sort cannot is not stable as it returns a sorted list of wavelengths and does not output indices. Thus Rainbow Sort is an instance of the Restricted Model.

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 (9) and the Model triple $S = (m, p, 0)$ from Section 3.4 we see that Generalised Rainbow Sort is an instance of the Model.

Implementation. Here we show an implementation of Generalised Rainbow Sort. A list of numbers to be sorted $L = (635, 592, 513, 426, 513, 592, 426, 635, 426, 513, 592)$ was encoded as beams of light of a proportional wavelength. The beams were arranged in order, parallel to the axis face of a prism (see Figure 6). The prism refracted the longer wavelengths to a lesser degree and so ordered them according to wavelength as shown in Figure 7.

Reading off the beams in order of the most refracted and in the order of the list L yields the list of indices $c(L) = (4, 7, 9, 3, 5, 10, 2, 11, 1, 6, 8)$. This resulting list of numbers is a stable sorting of the list L.

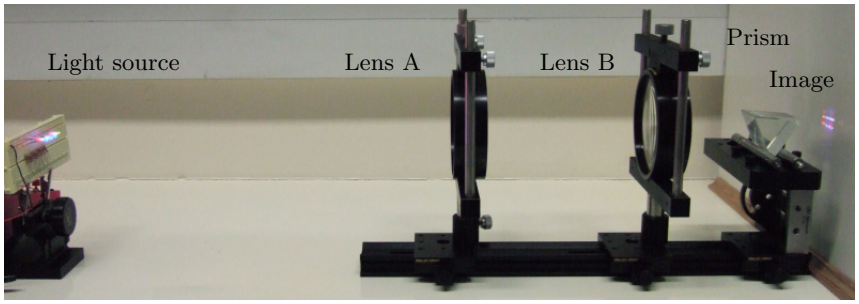

Fig. 6. The apparatus used to implement Generalised Rainbow Sort. Light emitting diodes were used as the light source. Lenses were used to focus the light beams onto the prism.

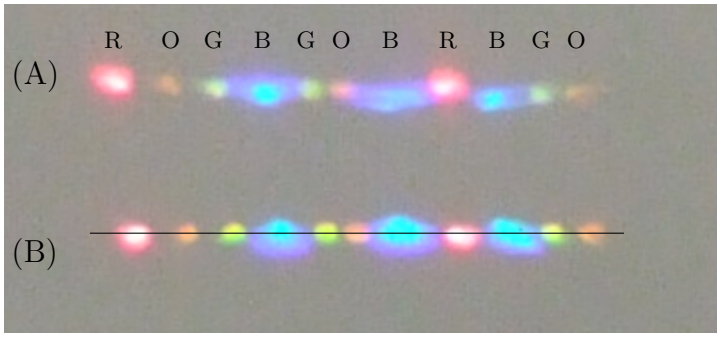

Fig. 7. (A) Beams that have been refracted proportional to their wavelength. Beams that are lower down than others have a lesser wavelength than those higher up. (B) Unrefracted light and the status of the beams before they were refracted by the prism. There is some distortion in the image introduced by the lenses. The coloured points are labeled G for green, O for orange, B for blue, and R for red.

3.5 Mass Spectrometry Sort

Mass spectrometry [15] is a technique used for separating ions by their mass-to-charge ratio and is most commonly used to identify unknown compounds and to clarify the structure and chemical properties of molecules. Of the several types of mass spectrometry we describe here the "time of flight" method.

Description. The process of time of flight mass spectrometric analysis is as follows. The gaseous sample particles are ionised by a short pulse of electrons and accelerated to a speed that is inversely proportional to their mass and directly proportional to their charge by a series of high voltage electric fields towards a long field-free vacuum tube known as the field-free drift region.

Here each ion moves at its entry velocity as they travel along the vacuum tube in a constant high voltage. At the opposite end of the tube there is a detector to record the arrival of the ions. In reflection time-of-flight mass spectrometers there is a "mirror" electric field which reflects the ions back along the length of the tube to the detector. This compensates for the initial energy spread of the ions and provides increased mass resolution. Since the different ions all travel the same distance but with characteristic velocities they arrive at the detector at different times. Using the time of arrival (time of flight) we identify the ions.

Sorting. We refer to the use of mass spectrometry for sorting as Mass Spectrometry Sort. Given a list L of numbers to be sorted, we encode each element of L as a sample of molecules with a time of flight proportional to the element value. The samples of molecules are fired simultaneously by a mass spectrometer. The time of flight of each element is then recorded as it arrives at the sensor. This is the sorted list. Since we cannot record or distinguish multiple instances in the input Mass Spectrometry Sort is an unstable sort.

Our usual technique for introducing stability is to run identical instances that sort each element in parallel. However due to the cost of a mass spectrometer this is unfeasible.

Instance. Gross [15] provides a simplified equation to describe the time of flight t of an ion based on its mass-to-charge ratio m_i/z is as follows

$$t = \frac{s}{\sqrt{2eU}} \sqrt{\frac{m_i}{z}} + t_0 \tag{10}$$

Where s is the distance traveled in the field-free region, U is the voltage of the field-free drift region, e is the charge of an electron and t_0 is a time offset. Equation (10) satisfies Equation (1) if we let $S = (m, \frac{s}{\sqrt{2eU}}, t_0)$ where m is the minimum time of flight that our apparatus measures. Our usual technique of running several apparatuses in parallel to achieve stability in the sorting is not practical due to expense. So in the way we have described it, Mass Spectrometry Sort is an instance of the Restricted Model of Physical sorting that is given in Definition 5.

Implementation. As an example of Mass Spectrometry Sort we use the data from an existing mass spectrum [12] and interpret it as an instance of Mass Spectrometry Sort. If we are presented with an unordered list of integers $T = (19, 24, 9, 13, 19, 9, 14, 13)$ and we know the root mass-to-charge ratio for enough ions we map the values of each element of T to a root mean charge ratio. In this case we map the ion CH_3^+ to 9, CHO^+ to 13, CH_3O^+ to 14, CH_3OH^+ to 14, $(CH_3OH)_2^+$ to 19 and $(CH_3OH)_3^+$ to 24. The compounds necessary to create these ions are then placed in the spectrometric apparatus and ionised. The order that the compounds arrive at the detector is the sorted list and is shown in Figure 8.

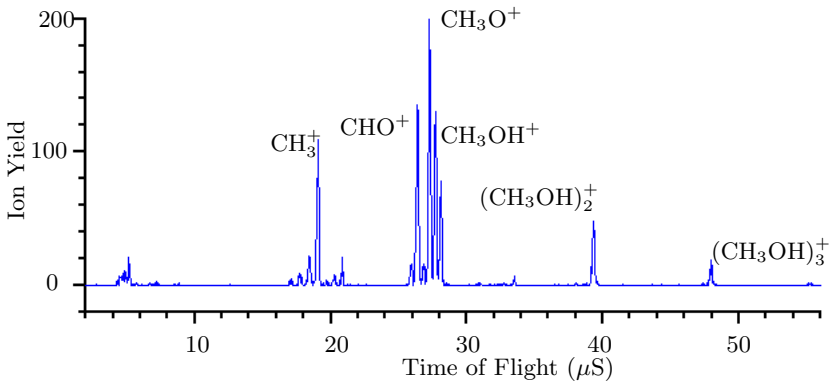

Fig. 8. A mass spectrum [12] representing the sorting of a list. Unlabeled peaks are not involved in the sort.

The time of arrival of each ion indicates the ion's identity and by reading the spectrograph the ordered list is found. In this case the sorted list is $c(T) = (19, 26, 27, 28, 39, 48)$ and since duplicates in the list have been lost it is clear that this is a non-stable sort.

4 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. We also define a Restricted Model of Physical Sorting which is an unstable sort.

We have provided five physical instances that are well-known laboratory techniques from experimental science as examples of physical sorts that are instances of the Model. We showed how the relationship between the Model and the Restricted Model naturally suggests how to introduce stability into a particular

existing physics-inspired sort, that being Rainbow Sort. In four of the five instances we have presented experimental implementations. Several of the implementations has the potential to sort millions or more items in constant time. We assume however, the processes of encoding the input values and reading off the results are linear time bottlenecks. It might be possible that these bottlenecks be mitigated by parallelising these operations. Other candidate instances of the model that we have not considered here are centrifugal separation and fractional distillation [20].

There are many other sorting techniques available in the literature. All sequential comparison-based sorts have a lower bound of $\Omega(n \log n)$ comparisons [17] and some sequential non-comparison based sorting algorithms such as Radix Sort [17] and Counting Sort [17] have a worst case of $O(n \log n)$ time. Constant time parallel sorting is possible, however an unfeasible number of processors is required [13]. An approach from optical computing provides a constant time sort using a combination of lenses and individual sensors with on-board processing [18]. Other nature inspired sorting techniques such as Bead Sort [3] and Spaghetti Sort [2] are related to our Model of Physical Sorting and also have a linear time bound on sequentially reading the result. Several of our suggested implementations have the advantage that their technologies are already currently being used in research laboratories worldwide.

The practical advantages of our Model of Physical Sorting over traditional sorting algorithms would certainly not be apparent until the amount of objects to be sorted is in the billions. Even then, the implementation prospects of the Model could be in doubt. However, there are two reasons why we think the possibility should not be discounted completely. Firstly, several of the laboratory techniques underlying the implementations in this paper are widely used for other scientific purposes and significant resources are invested annually to improving the accuracy and reliability of the technologies. Each of these advances would benefit a sorting apparatus implemented using the same technology. Secondly, where an individual wishes to sort a set of particles by some property, it might be more efficient to directly use one of the technologies described in this paper rather than individually sensing the appropriate property of each particle, transferring it to a computer, and performing a traditional sort. The practicality of the Model of Physical sorting would also be less in doubt if there were implementations with a small footprint (for example 30cm^2) and were easily reusable.

5 Acknowledgments

This work was partially funded by the Irish Research Council for Science, Engineering and Technology. We would like to thank Charles Markham, William Lanigan, and Fiachra Matthews for their help in implementing Generalised Rainbow Sort. We would also like to thank reviewers for their helpful comments.

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, June 1984.
3. Joshua J. Arulanandham, Cristian S. Calude, and Michael J. Dinneen. Beadsort: A natural sorting algorithm. *The Bulletin of the European Association for Theoretical Computer Science*, 76:153–162, 2002.
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):841–856, 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. Pei Yu Chiou, Aaron T Ohta, and Ming C Wu. Massively parallel manipulation of single cells and microparticles using optical images. *Nature*, 436:370–372, July 2005.
10. Isaac L. Chuang, Neil Gershenfeld, and Marc Kubinec. Experimental implementation of fast quantum searching. *Physical Review Letters*, 18(15):3408–3411, 1998.
11. Benjamin De Lacy Costello and Andrew Adamatzky. Experimental implementation of collision-based gates in Belousov-Zhabotinsky medium. *Chaos, Solitons and Fractals*, 25(3):535–544, August 2005.
12. Elaine M. Duffy. Studying a supersonic beam of clusters using a mass spectrometer. Master's thesis, National University of Ireland, Maynooth, jul 2005.
13. William Gasarch, Evan Golub, and Clyde Kruskal. A survey of constant time parallel sorting. *Bulletin of the European Association for Theoretical Computer Science*, 72:84–103, 2000.
14. Jesper Glückstad. Microfluidics: Sorting particles with light. *Nature Materials*, 3:9–10, 2004.
15. Jürgen H. Gross. *Mass Spectrometry: A Textbook*. Springer, 2004.
16. 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.
17. Donald E. Knuth. *The Art of Computing Programming: Sorting and Searching*, volume 3. Addison-Wesley, second edition, 1998.
18. Ahmed Louri, James A. Hatch, Jr., and Jongwhoa Na. A constant-time parallel sorting algorithm and its optical implementation using smart pixels. *Applied Optics*, 34(17):3087–3097, 1995.
19. Carver Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, Massachusetts, 1989.
20. Clifton E. Meloan. *Chemical Separations: principles, techniques, and experiments*. Wiley-Interscience, 1999.

21. Thomas Naughton, Zohreh Javadpour, John Keating, Milos Klíma, and Jirí Rott. General-purpose acousto-optic connectionist processor. *Optical Engineering*, 38(7):1170–1177, 1999.
22. Colin F. Poole and Sheila A. Schuette. *Contemporary practice of chromatography*. Elsevier, 1984.
23. Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
24. Dominik Schultes. Rainbow sort: Sorting at the speed of light. *Natural Computing*, 5(1):67–82, 2006.
25. Colin F. Simpson and Mary Whittaker, editors. *Electrophoretic techniques*. Academic Press, London, 1983.
26. 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.
27. 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.
28. Damien Woods and Thomas J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1–3):227–258, April 2005.