The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Skyline Computation for Low-Latency Image-Activated Cell Identification

Kenichi Koizumi, Kei Hiraki, Mary Inaba
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan +81-3-5841-4110
{koiken, hiraki, mary}@is.s.u-tokyo.ac.jp

Real-Time Artificial Intelligence Research

Because of breakthroughs in the field of deep learning, the accuracy of image classification and multimedia recognition in Artificial Intelligence (AI) research has improved rapidly. In this study, the objective is the classification of high-dimensional data, and, in particular, the screening of very rare entries from a large population. In general, the initial set of vectors is divided into several groups by using a clustering method, and an outlier detection method identifies distinct entries such as noise. Our focus is on a new cognitive research problem in which the aim is to detect *boundary* entries, namely entries that are geometrically located near the outer edges of a population in multidimensional space.

Serendipiter (Guo et al. 2017) is a fast cell sorter that discovers very rare cells with atypical ability from an enormous number of cells. A block diagram of Serendipiter is shown in Figure 1. The cells are identified from the cell-measurement information obtained by multiple sensor technologies, such as optical imaging and spectroscopy. The measurement and identification latencies must be below ten milliseconds. The discovery of very rare cells (constituting one per a trillion cells) within a realistic time is anticipated for efficient biofuel production by *Euglena* spp. and high-precision blood testing. Serendipiter accurately analyzes single cells and classifies 10,000 cells per second. We approached this challenge by using the skyline computation method and considering the set of boundary entries as a set of skyline points.

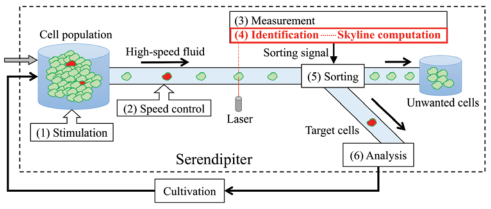

Figure 1: Block diagram of Serendipiter

Skyline Computation

The skyline computation algorithm (Börzsönyi, Kossmann, and Stocker 2001) is used to extract extraordinary entries

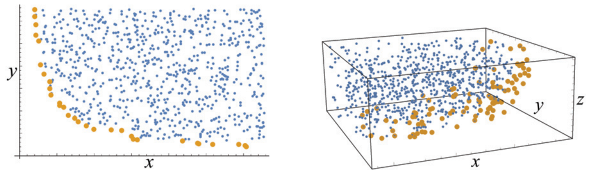

Figure 2: Examples of skylines in multidimensional spaces (left: two dimensions and right: three dimensions); the yellow and blue points denote the skyline and non-skyline points, respectively.

from a database containing entries with multiple attributes. Points that are not dominated by the other points are called *skyline* or *pareto-optimal* points. The left and right panels of Figure 2 show examples of skylines in two- and three-dimensional space, respectively.

The task of computing the skyline of a *dynamic* set of points is known as the *continuous skyline computation* (Morse, Patel, and Grosky 2007). Real databases are updated frequently in streaming or other suitable environments. The injection and ejection times of an entry into and from the database are called the *activation* and *deactivation* times of the entry, respectively. In a continuous skyline computation, a set of skyline points is updated due to point activation and deactivaton.

The JR-tree

The continuous skyline computation presents a difficult problem because the activation or deactivation of just one point can demote many existing skyline points from the skyline, or promote many existing points to the skyline, both of which will change a substantial part of the skyline. Previous methods utilize spatial indexing trees, such as B-trees, R-trees, or quadtrees (Morse, Patel, and Grosky 2007). The JR-tree (Koizumi, Inaba, and Hiraki 2015) is the state-of-the-art algorithm for continuous skyline computation. A *complete dominance graph* is a directed acyclic graph, whose vertex represents each point and whose arc represents each dominance relation. A JR-tree is a rooted spanning subtree of the complete dominance graph. The JR-tree algorithm is faster than other algorithms because (1) it uses a more appropriate hierarchical structure, and (2) it is dimension-independent.

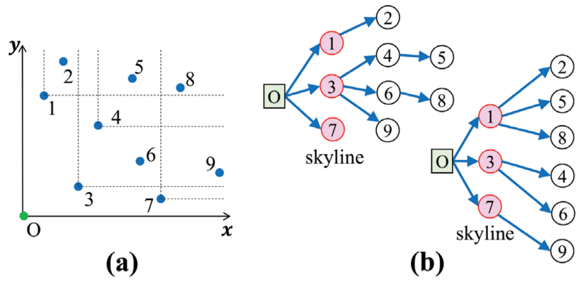

Figure 3: Two examples of JR-trees (b) constructed from nine points represented by two-dimensional vectors (a).

Figure 3 shows two examples of JR-trees.

To output the skyline, we need to enumerate the children of O, which corresponds to the origin; but there is no need to traverse the tree. In continuous skyline computation, the injection and ejection operations generate one possible JR-tree that represents the current set of points. The injection increases the depth of the tree. However, the computation of those points far from the origin (such as deeper vertices) can be delayed because their results may not be required until the end of the computation. To postpone such an operation, we make use of lazy injection. Based on JR-tree, we propose a hardware algorithm for high-speed continuous skyline computation and implement it on a field-programmable gate array (FPGA). Figure 4 shows the logic diagram of its design.

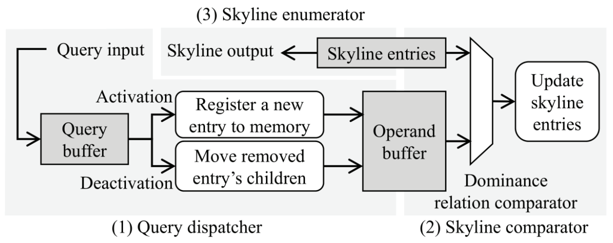

Figure 4: Logic diagram for the hardware implementation of the JR-tree, consisting of (1) Query dispatcher, (2) Skyline comparator, and (3) Skyline enumerator.

Evaluation

We generated three types of random synthetic dataset, **CORR**elated, **INDE**pendent, and **ANTI**-correlated, characterizing the spatial information by the respective correlation strengths of the spatial distributions.

We also extracted features from cell images taken with a fluorescence microscope, which is high-resolution images from the CYTO 2017 Image Analysis Challenge (CYTO 2017), and created real-world datasets. The images were taken with a Leica SP5 confocal fluorescence microscope. They display immunostaining of human proteins and consist of four fluorescence channels (see Figure 5). Each image was tagged using 19 types of labels. We created two datasets that consisted of 19,495 three-dimensional entries

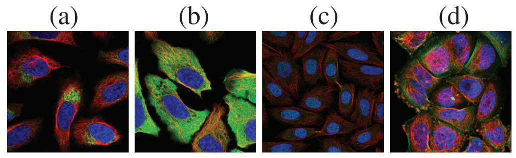

Figure 5: Examples of human protein cell images from the CYTO 2017 Image Analysis Challenge. Each image is tagged as follows: (a) Mitochondria, (b) Cytosol, (c) Cytokinetic Bridge, and (d) Focal Adhesion Sites.

called CYTOSOL and FAS. We allocated uniformly distributed lifetimes to the entries to create datasets for continuous skyline computation.

We evaluated our proposed algorithms against two existing software algorithms for continuous skyline computation using both synthetic and real-world datasets (see Figure 6). For the synthetic datasets, the hardware implementation of the JR-tree was the fastest method; in particular, for the five-dimensional CORR datasets, it was 3.7 times faster than the software implementation of the JR-tree. As the distribution's correlation coefficient increased, the hardware-based JR-tree became faster. For the CYTOSOL dataset, the hardware JR-tree was 1.7 times faster than the software JR-tree.

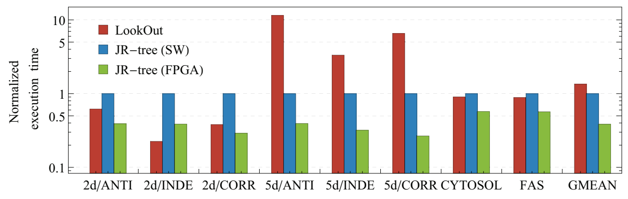

Figure 6: Comparison of the normalized execution times for the existing software implementations, LookOut and our software and hardware implementations of the JR-tree.

References

Börzsönyi, S.; Kossmann, D.; and Stocker, K. 2001. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, 421–430. IEEE.

CYTO. 2017. Image analysis challenge.

Guo, B.; Lei, C.; Kobayashi, H.; Ito, T.; Yalikun, Y.; Jiang, Y.; Tanaka, Y.; Ozeki, Y.; and Goda, K. 2017. High-throughput, label-free, single-cell, microalgal lipid screening by machine-learning-equipped optofluidic time-stretch quantitative phase microscopy. *Cytometry Part A*.

Koizumi, K.; Inaba, M.; and Hiraki, K. 2015. Efficient implementation of continuous skyline computation on a multi-core processor. In *Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on*, 52–55. IEEE.

Morse, M.; Patel, J. M.; and Grosky, W. I. 2007. Efficient continuous skyline computation. *Information Sciences* 177(17):3411–3437.