Local Search in Histogram Construction

Felix Halim and **Panagiotis Karras** and **Roland H. C. Yap**
School of Computing
National University of Singapore
13 Computing Drive
Singapore 117417
{halim, karras, ryap}@comp.nus.edu.sg

Abstract

The problem of dividing a sequence of values into segments occurs in database systems, information retrieval, and knowledge management. The challenge is to select a finite number of boundaries for the segments so as to optimize an objective error function defined over those segments. Although this optimization problem can be solved in polynomial time, the algorithm which achieves the minimum error does not scale well, hence it is not practical for applications with massive data sets. There is considerable research with numerous approximation and heuristic algorithms. Still, none of those approaches has resolved the quality-efficiency tradeoff in a satisfactory manner. In (Halim, Karras, and Yap 2009), we obtain near linear time algorithms which achieve both the desired scalability and near-optimal quality, thus dominating earlier approaches. In this paper, we show how two ideas from artificial intelligence, an efficient *local search* and recombination of multiple solutions reminiscent of *genetic algorithms*, are combined in a novel way to obtain state of the art histogram construction algorithms:

Introduction

The problem of *histogram construction* or *sequence segmentation* has numerous applications in database systems (Ioannidis 1993; Chakrabarti et al. 2001), decision support (Acharya et al. 1999; Ioannidis and Poosala 1999), bio-informatics (Salmenkivi, Kere, and Mannila 2002), and information retrieval (Chakrabarti et al. 2002). (Ioannidis 2003) summarizes its long history and extensive literature.

The histogram construction problem is to divide a sequence of values into a number of piecewise-constant line *segments* (*buckets*), approximating each segment by a single representative to achieve low *approximation error*. The most common error metric is the Euclidean, or *root-mean-square* error. A histogram that minimizes this error turns out to be the *optimal* choice when approximating attribute value frequencies for estimating the query result size (Ioannidis and Poosala 1995) in databases. The optimization problem to minimize the Euclidean error is solvable with an $O(n^2B)$ dynamic-programming algorithm, V-Optimal (Bellman 1961; Jagadish et al. 1998). However, it is not scalable enough to be used in practice; thus, real-world database systems employ simple heuristics instead (Ioannidis and Poosala 1995; Poosala et al. 1996).

Recently, elaborate approximation algorithms have been devised; these either provide a computational shortcut by approximating the error function itself (Guha, Koudas, and Shim 2006) (AHistL-Δ), or employ a divide-and-conquer approach which merges the sub-solutions to give error guarantees (Terzi and Tsaparas 2006) (DnS). Ideally, such algorithms should achieve a near-linear time efficiency and near-optimal quality. However, in practice, they fall short of both these goals. Their running time approaches, and can even exceed, that of V-Optimal, with larger errors.

In (Halim, Karras, and Yap 2009), we showed some state of the art histogram construction algorithms which were not only efficient and scalable but achieve good solution quality. We achieved substantial improvement in the middle ground between the robustness of approximation algorithms and the simplicity of heuristics. Although the algorithms provide no approximation guarantees, our solutions combine scalability and quality in a way that other approaches do not; thus, they are inherently suited for the segmentation of very large data sets. We experimentally demonstrate that our algorithms vastly outperform guarantee-providing schemes in running time, while achieving comparable or superior approximation accuracy.

In this paper, we summarize the results and explain how ideas from *local search* and *genetic algorithms* were instrumental in these novel algorithms. By using ideas from artificial intelligence rather than following the approach in the database literature, we were able to improve either on the running time or accuracy compared to existing algorithms. The approach we used may also lead to new and improved algorithms for problems where a combination of local search to get good and diverse solutions is combined with an efficient polynomial time improvement/optimization procedure.

V-Optimal Histogram Construction

Given an n-length data vector $\mathbf{D} = \langle d_0, d_1, \ldots, d_{n-1} \rangle$, the problem is to find a piecewise constant representation $\hat{\mathbf{D}}$ of $\mathbf{D}$ using at most B segments that minimizes the *Euclidean error*, $\mathcal{L}_2(\hat{\mathbf{D}}, \mathbf{D}) = \left(\frac{1}{n}\sum_i |\hat{d}_i - d_i|^2\right)^{\frac{1}{2}}$, where $\hat{d}_i$ denotes the approximate estimated value for d_i. In practice, it suffices to minimize the sum of squared errors (SSE),

$\sum_i |\hat{d}_i - d_i|^2$. We remark that while there is work to generalize the problem to wider classes of error metrics, $\mathcal{L}_2$ remains an important error metric for several applications (Ioannidis and Poosala 1995; Himberg et al. 2001; Chakrabarti et al. 2002) which remains our focus.

The approximate representation of $\mathbf{D}$ is called a *histogram*, a *segmentation* (Terzi and Tsaparas 2006), a *partitioning*, or a *piecewise-constant approximation* (Chakrabarti et al. 2002). The histogram, $\mathcal{H}$, divides $\mathbf{D}$ into $B \ll n$ disjoint intervals (*buckets*, *segments*) $[b_i, e_i]$, where b_i and e_i are *indices* of data items and $1 \leq i \leq B$. A single representative value v_i approximate all values d_j in a bucket, $j \in [b_i, e_i]$. The value v_i that minimizes an error metric in a bucket is defined as a function of the data therein (Terzi and Tsaparas 2006; Karras, Sacharidis, and Mamoulis 2007; Guha, Shim, and Woo 2004); for the $\mathcal{L}_2$ metric, the optimal value of v_i is the *mean* of values in $[b_i, e_i]$ (Jagadish et al. 1998). A *segmentation algorithm* aims to define boundary positions $e_1, \ldots, e_{B-1}$ that achieve a low overall error.

An $O(n^2B)$ algorithm that constructs an $\mathcal{L}_2$-optimal segmentation along the lines of (Bellman 1961) was presented by (Jagadish et al. 1998) and improved by (Guha 2008). It recursively derives the optimal (i.e., SSE-minimal) b-segmentation of $\langle d_0, d_1, \ldots, d_i \rangle$ from the optimal $(b-1)$-segmentations of its prefixes, expressing its SSE as:

$$E(i, b) = \min_{b \leq j < i} \{E(j, b-1) + \mathcal{E}(j+1, i)\} \qquad (1)$$

$\mathcal{E}(j+1, i)$ is the minimal SSE for the segment $\langle d_{j+1}, \ldots, d_i \rangle$, computed in $O(1)$ using pre-computed sum quantities (Jagadish et al. 1998). However, the quadratic complexity of V-Optimal renders it inapplicable in most real-world applications, thus various approximation schemes have been proposed.

Approximation Algorithms

(Terzi and Tsaparas 2006) suggested a sub-quadratic factor-3 approximation algorithm as an alternative to V-Optimal. Their *divide and segment* (DnS) algorithm arbitrarily partitions the problem sequence into smaller subsequences, segments each of those optimally (employing a V-Optimal subprocess), and then combines (i.e., locally merges) the derived segments into B final buckets, treating them as the individual elements in another application of V-Optimal. Assuming that the original sequence is partitioned into $\chi = \left(\frac{n}{B}\right)^{2/3}$ equal-length segments in the first step, then DnS has a worst-case complexity of $O(n^{4/3}B^{5/3})$.

(Guha, Koudas, and Shim 2006) suggest AHistL-Δ, an $O(n + B^3(\log n + \epsilon^{-2}) \log n)$-time algorithm that computes an $(1+\epsilon)$-approximate B-bucket histogram. As $E(j, b-1)$ in Equation 1 is a non-decreasing function of j, $E(j, b)$ can be approximated by a *staircase* histogram, where the value at the right-hand end of a segment is at most $(1+\delta)$ times the value at the left-hand end, with $\delta = \frac{\epsilon}{2B}$.

Existing Greedy Heuristics

Out of several simple greedy heuristics for histogram construction (such as the end-biased (Ioannidis and Poosala 1995), equi-width (Kooi 1980), and equi-depth (Piatetsky-Shapiro and Connell 1984; Muralikrishna and DeWitt 1988) heuristics), two stand out: The former, MaxDiff, advocated as "probably the histogram of choice" by (Poosala et al. 1996), selects the $B-1$ points of highest difference between two consecutive values as boundary points in $O(n \log B)$ time. The latter, MHIST, repeatedly selects and splits the bucket with the highest SSE, making B splits (Poosala and Ioannidis 1997; Jagadish et al. 1998) in $O(B(n + \log B)$ time. (Jagadish et al. 1998) observe that MaxDiff performs better on spiked data and MHIST on smooth data.

Fast and Effective Histogram Construction

An ideal segmentation algorithm should provide a satisfactory tradeoff between efficiency and accuracy, thus providing a significant advantage with respect to both the optimal but not scalable V-Optimal and to fast but inaccurate heuristics. Approximation schemes with time super-linear in n and/or B may not achieve this goal. We propose alternative schemes based on local search. Although our local search and existing heuristics are both greedy algorithms, there are important differences. Our local search algorithms employ iterative improvement and stochasticity. In contrast, both MaxDiff and MHIST derive a solution in one shot and never modify a complete B-segmentation they have arrived at.

A Basic Local Search Algorithm: GDY

We first describe a basic local search algorithm called GDY which can already be quite effective compared with greedy heuristics. It starts with an *ad hoc* solution (a segmentation) $\mathcal{S}_0$ and makes local moves which greedily modify boundary positions so as to reduce the total $\mathcal{L}_2$ error. Each local move has two components. First a segment boundary whose removal incurs the *minimum* error increase is chosen. As this decreases the number of segments by one, a new segment boundary is added back by splitting the segment which gives the *maximum* error decrease. Note that expanding or shrinking a segment by moving its boundary is a special case of this local move when the same segment is chosen for removal and splitting. A local minimum on the total histogram error is reached when no further local moves are possible. Figure 1 gives the GDY algorithm.

To ensure efficient local moves, we keep a min-heap H^+ of *running* boundary positions with their associated potential *error increase*, and a max-heap H^- of all *running* segments with the potential *error decrease*. The time complexity of GDY is $O\left(M\left(\frac{n}{B} + \log B\right)\right)$, where M is the number of local moves.

Improving with Local Search Sampling: GDY_DP

Experiments show that the basic local search GDY algorithm already performs quite well and efficient. We now use GDY as a building block for two algorithms (GDY_DP and GDY_BDP) which achieve much lower total error without sacrificing the performance.

We observe that one run of GDY often finds at least 50% of the optimal partitions. Typically, a local search strategy

Algorithm GDY(B)
Input: space bound B, n-data vector $\mathbf{D} = [d_0, \ldots, d_{n-1}]$
Output: histogram $\mathbf{H}$ of B segments
1. $S_0 =$ initial (random) $B-1$ segment boundaries on $\mathbf{D}$;
2. $i = 0$; Populate H^+ and H^- with S_i;
3. **while** (H^+ is not empty)
4. $i = i + 1$; $S_i = S_{i-1}$;
5. G = extract boundary with minimum $\Delta^+ E_i$ from H^+;
6. Remove G from S_i and update H^+ and H^-;
7. P = choose segment with maximum $\Delta^- E_j$ from H^-;
8. **if** $\left(\Delta^+ E_i - \Delta^- E_j \geq 0\right)$
9. Undo steps 6 and 7; // G is discarded from H^+
10. **else** Split segment P, add new boundary to S_i;
12. Update partitions' costs in H^+ and H^-;
13. **return** S_i;

Figure 1: GDY algorithm

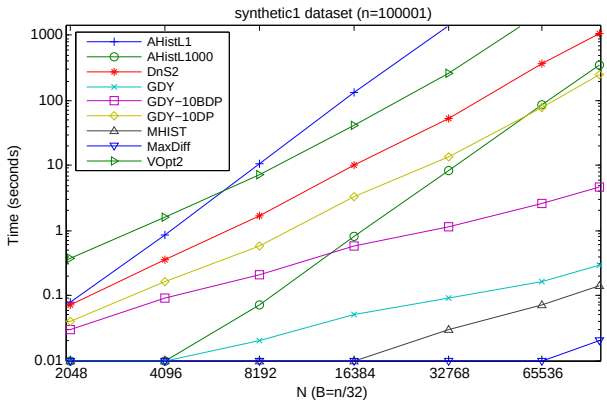

Figure 3: Runtime vs. n, $B = \frac{n}{32}$: Synthetic

employs restart to increase the percentage of optimal partitions. Instead, we use a more effective strategy which relies on the fact that V-Optimal gives the optimum solution but is quadratic in n.

We first deal with the case for local search segmentation when $B \leq \sqrt{n}$, the GDY_DP algorithm. Our approach is inspired by the recombination/crossover operator and fitness selection from genetic algorithms (GA). Since a single run of GDY already gives a good segmentation (we found it experimentally to be superior to MaxDiff and MHIST), we make I local search runs, merging all partitions from the solution in each run into a single candidate sample set. We expect that a large percentage of the partitions in the candidate sample set are also in an optimal solution. Since the sample set has at most IB partitions, it is still $O(B)$, as such selecting the best B out of the set using dynamic programming costs $O(B^3)$ which gives a total runtime of $O(nB)$ as $B \leq \sqrt{n}$. Figure 2 gives the GDY_DP algorithm.

Algorithm GDY_DP(B, I)
Input: bound B, number of runs I, n-data vector $\mathbf{D}$
Output: histogram partitioning $\mathbf{H}$ of B segments
1. $\mathcal{S} =$ empty set of sample boundaries
2. **loop** (I times)
3. $\mathcal{P} =$ GDY(B); // run GDY with random initialization
4. $\mathcal{S} = \mathcal{S} \cup \mathcal{P}$;
5. **return** V-Optimal$(\mathcal{S}, B)$;

Figure 2: GDY_DP algorithm

A Batch-Processing Variant

We now consider the case when $B > \sqrt{n}$. Now, we can no longer use V-Optimal to recombine the samples to get a guaranteed improvement on the local search segmentation, while also maintaining a linear time complexity in n. This is because the time complexity of the V-Optimal step in GDY_DP becomes too large. We introduce the GDY_BDP algorithm, a batch-processing version of GDY_DP to handle this case.

When $B > \sqrt{n}$, to maintain scalability, we cannot have as many samples as in GDY_DP. The idea is to use V-Optimal to improve a subspace inside a solution. Here, a subspace consists of a contiguous number of segments within a solution. Thus, we employ the sampling idea in GDY_DP to improve a subpart of a solution. As in GDY_DP, subspaces of local search solutions from several runs will often be part of an optimal solution.

We start with an *auxiliary* solution $\mathcal{A}$ from one run of GDY. Samples are then collected with multiple runs of GDY into a set of segments $\mathcal{S}$. Starting from one end of solutions collected in $\mathcal{S}$, $\sqrt{n}$ consecutive segments $\bar{s}$ are picked from $\mathcal{S}$ forming a *batch*. In auxiliary solution $\mathcal{A}$, we select the smallest subspace $\bar{a}$ enclosing $\bar{s}$. Now V-Optimal can be employed to find a better segmentation of size $|\bar{a}|$ from $\bar{s} \cup \bar{a}$ which revises the corresponding subspace in $\mathcal{A}$. An important point is that unlike DnS, our sampling process is non-uniform and results in more samples at subspaces where more segments may be needed, thus reducing error. The next batch of $O(\sqrt{n})$ samples is processed similarly, and so on. Due to lack of space, we refer the reader to the GDY_BDP algorithm in (Halim, Karras, and Yap 2009). The result is that GDY_BDP takes $O(nB)$ time, scaling well in both the input size n and the number of segments B, and produces similar quality to that of GDY_DP. While GDY_DP is reminiscent of a GA with a single recombination step, GDY_BDP performs multiple, incremental recombinations on subspaces of auxiliary solution $\mathcal{A}$.

Experimental Evaluation

We conducted an extensive experimental comparison of approximation, heuristic and our algorithms on real-world time series data sets.[1] Our experiments show that our local search algorithms are both scalable and fast, and also of high quality, often almost the same as V-Optimal. Here, we highlight the scalability versus quality tradeoffs. Figure 3 plots runtime growth with respect to n when $B = \frac{n}{32}$, on logarithmic axes, with a synthetic data set. Neither AHistL-Δ, nor V-Optimal, nor DnS scale well while GDY_BDP presents a runtime growth trend comparable to that of simple heuristics.

[1] See (Halim, Karras, and Yap 2009) and `http://felix-halim.net/histogram` for details.

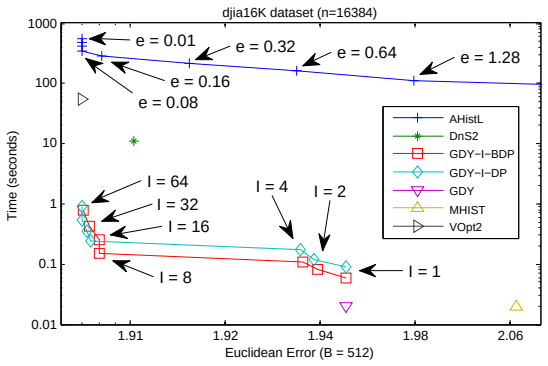

Figure 4: Tradeoff Delineation, $B = 512$: DJIA

A fast algorithm is useless unless it gives good quality. Figure 3 shows that our local search algorithms are more scalable than the non-heuristic algorithms with the heuristics ones being faster. Turning to quality gives a completely different picture. Figure 4 shows error versus time (log scale) on a Dow Jones time series data set. Variants of AHistL-Δ for different ϵ get their own dot. Lower values of ϵ allow for higher accuracy at the price of extra runtime. Likewise, several variants of GDY_DP and GDY_BDP are presented, based on the number I of iterations of GDY used for collecting samples. The graph confirms that AHistL-Δ performs poorly at resolving the quality-efficiency tradeoff. An attempt to gain quality by lowering ϵ renders the runtime higher than that of V-Optimal; an effort to improve time-efficiency by increasing ϵ is not effective, while losing quality. DnS does not fare much better. In contrast, GDY_DP and GDY_BDP give the best resolution of the tradeoff, as they dominate the other algorithms either in time or quality or both. Our algorithms allow tuning the sample size and algorithm choice to trade off runtime cost against quality.

Discussion

Our study has led to some remarkable findings. First, despite their elegance, approximation schemes with robust error guarantees do not achieve an attractive resolution of the tradeoff between efficiency and quality. Among the proposed schemes, DnS achieves a slightly better tradeoff than AHistL-Δ. Worse still, both can be superseded in time efficiency by V-Optimal, whom they are meant to approximate, due to their super-linear dependence on B (detailed in (Halim, Karras, and Yap 2009)). Secondly, by employing a local search that exploits the optimal dynamic programming segmentation and sampling, we address the tradeoff much more satisfactorily. Our best performing algorithm, GDY_BDP, consistently achieves near-optimal quality in near-linear time.

In developing a good local search algorithm, one needs to be creative in designing heuristics to guide the local moves. Such local search tuning is known to be tedious and very time consuming and often requires an involved experimentation and evaluation process. Furthermore, the approaches taken are customized to a particular instance of the problem, which makes it hard to generalize to other unrelated domains.

We present an alternative improvement strategy for problems where there is an efficient (and in our case, optimal) algorithm for a sub-problem. In this case, local search is used to collect a diversified set of solutions. One has to devise the local search to give sufficiently good quality samples(which GDY achieves). The efficient improvement algorithm can improve part of the solution using the collected samples from local search. Thus, we show that local search can be effective not just on intractable problems but also in polynomially solvable problems that present a premium in terms of efficiency. Such problems arise frequently in the fields of data mining and data engineering. We show that the "unreasonable effectiveness of local search" also succeeds here. Furthermore, any improvements/tuning/new heuristics in the local search would likely automatically translate to improvements in the overall algorithm as it could be employed to get better samples or run more efficiently.

Our solution employs a novel local search which maintains a population of solutions and uses a recombination of those solutions. It also exploits careful use of an optimal polynomial time optimization procedure. We think that the strategy used here may also be applicable to other problems with pseudo-polynomial optimization algorithms. Such a strategy may allow for effective scalable algorithms that exploit a synergy between local search and an optimal, but more expensive, pseudo-polynomial time algorithm.

To understand the difference between our approach and more typical local search approaches, we compare GDY_DP against a pure stochastic local search version, GDY_LS, which runs GDY runs for I iterations starting with a random initial partition. The difference between the two is that GDY_LS selects the best solution found within some iteration, while GDY_DP combines solutions which are reoptimized using V-Optimal. Our experiments use the same random seed, hence the i^{th} iteration in both GDY_LS and GDY_DP produce exactly the same i^{th} solution.

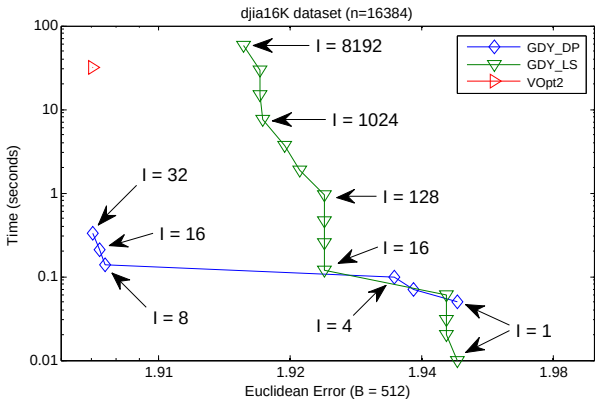

Figure 5: GDY_LS vs. GDY_DP, $B = 512$: DJIA

Figure 5 shows the performance of GDY_LS compared to GDY_DP. The pure local search, GDY_LS, takes far more iterations (hence, computation time) when compared

with GDY_DP which reaches the same quality as the optimal algorithm, VOpt2, in 32 iterations. We can see also that changing GDY_LS to a cleverer local search (perhaps utilizing some problem specific domain optimizations) can lower the number of iterations required, giving smaller times or lower error or both. The optimal algorithm thus serves as a catalyst for building variants of local search.

Visualization of the Search

In the foregoing study, we have assessed the degree to which a given histogram $\mathcal{H}$ approximates the optimal one $\mathcal{H}_o$ in terms of the Euclidean error metric. Another way of assessing the convergence towards the optimal histogram is to count the amount of bucket boundary positions that $\mathcal{H}$ and $\mathcal{H}_o$ do *not* have in common, or its *distance* to $\mathcal{H}_o$. The optimal solution is achieved when the distance is 0. The advantage of this distance metric is that it conveys an intuitive representation of how far a given histogram is from the optimal one, in a way that an error value does not.

In this Section, we present our evaluation of the algorithms we study in terms of the distance metric. Let a histogram $\mathcal{H}$ be expressed as the set of B boundary positions that define it. Then *distance* between two histograms $\mathcal{H}_1$ and $\mathcal{H}_2$ is defined as $d = B - |\mathcal{H}_1 \cap \mathcal{H}_2|$. A distance value d indicates that $\mathcal{H}_1$ has d boundaries that do not match any boundary in $\mathcal{H}_2$, and vice versa.

(Halim and Yap 2007; Halim, Yap, and Lau 2006) proposed a novel visualization, *Fitness Landscape Search Trajectory* (FLST), for visualizing the behavior of local search. We have adopted this technique to visualize the progress of GDY_DP as it collects samples from GDY runs. Figure 6 shows a two dimensional visualization which shows how different solutions from the different algorithms compare in terms of the distance to $\mathcal{H}_o$ as well as pair-wise distance among them. The visualization is intended to give an approximation to the distance between solutions – solutions with small d are close, while those with a larger d value are further apart. The indicated value of d is the distance of that solution compared to $\mathcal{H}_o$ (Vopt2) which is in the middle. The percentage difference from the optimal Euclidean error for each algorithm is also shown. The visualization depicts: two GDY_DP runs, for $I = 2$ and $I = 8$; eight different GDY runs; AHistL-Δ with $\varepsilon = 0.16$; and DnS. The runtime of each algorithm is presented in the lower graph which is aligned with the upper diagram and the time axis uses a logarithmic scale.

With respect to $\mathcal{H}_o$, it is noteworthy that each of the 8 GDY runs achieves distance of around 58-74 out of a maximum possible value of 512. This is a fairly good result, comparable to that of DnS. By exploiting only two different GDY runs, GDY_DP with $I = 2$ manages to decrease the distance to 32, which is similar to the distance achieved by AHistL-Δ. The visualization also shows that the solutions from different GDY are distributed around the graph which indicates that the GDY runs constitute a diverse set centered around $\mathcal{H}_o$. This diversity is needed in order for GDY_DP to be able to take advantage of a multitude of sample boundaries. As we have discussed, the diversity is not solutions being different due to randomness. Rather all GDY solutions,

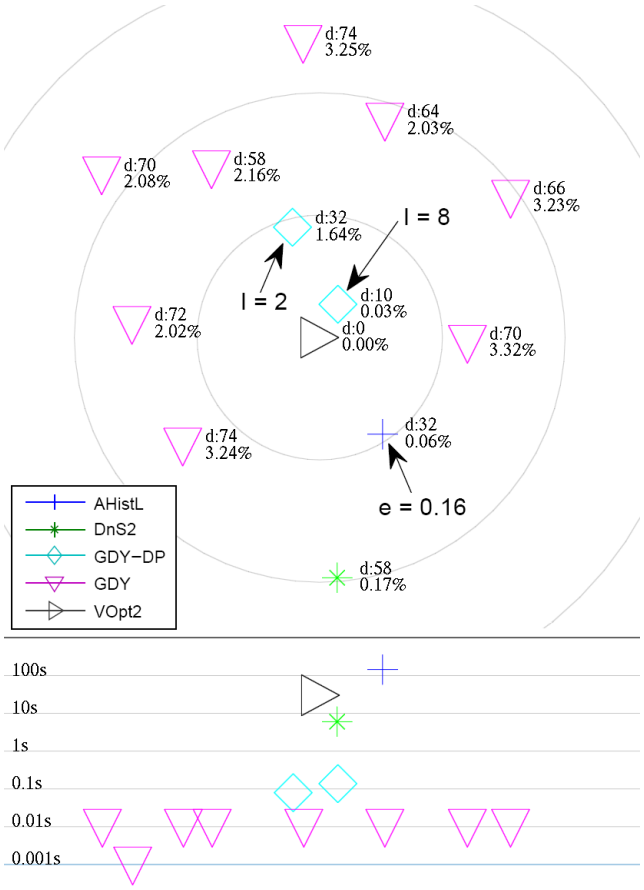

Figure 6: Comparing solution structure with quality and time, $B = 512$: DJIA

while different from each other, are valid solutions which are relatively close to the optimal one. We see that GDY_DP with $I = 8$ moves much closer to $\mathcal{H}_o$, the d value is one third of that for $I = 2$. Earlier, we already saw in Figure 4 that GDY_DP runs outperform DnS and AHistL $-\ \Delta$ error and runtime. Here, we see that one reason is because it shares more of the solution with $\mathcal{H}_o$. Thus, GDY_DP achieves a more attractive resolution of the quality/efficiency tradeoff. When $I = 32$, GDY_DP achieves the exact optimal solution.

Conclusion

In (Halim, Karras, and Yap 2009) we apply local search techniques to a classical problem of data engineering, namely histogram construction or sequence segmentation. Our approach addresses a critical gap in existing research. To the best of our knowledge, it is the *first* work to develop segmentation algorithms that are *both* fast and scalable (i.e., near-linear) in terms of time efficiency, *and* effective in terms of the quality they achieve. It is surprising that, for such a well studied problem, local search can make a significant impact against the previous state of the art. The key to our approach is the use of novel local search with some ideas borrowed from GAs, along with a careful use of a more

expensive (albeit polynomial time) optimization algorithm. We have conducted the *first*, to our knowledge, experimental comparison of proposed heuristics and approximation schemes for sequence segmentation. This study shows that our mixed approach achieves near-optimal quality in near-linear runtime, outperforming popular heuristics in quality and recently suggested approximation schemes in time efficiency as well as quality. In conclusion, our solutions provide a highly recommendable choice for all areas where segmentation or histogram construction finds application.

References

Acharya, S.; Gibbons, P. B.; Poosala, V.; and Ramaswamy, S. 1999. Join synopses for approximate query answering. In *ACM SIGMOD International Conference on Management of Data*, 275–286. New York, NY, USA: ACM Press.

Bellman, R. 1961. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* 4(6):284.

Chakrabarti, K.; Garofalakis, M.; Rastogi, R.; and Shim, K. 2001. Approximate query processing using wavelets. *VLDB Journal* 10(2-3):199–223.

Chakrabarti, K.; Keogh, E.; Mehrotra, S.; and Pazzani, M. 2002. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems* 27(2):188–228.

Guha, S.; Koudas, N.; and Shim, K. 2006. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems* 31(1):396–438.

Guha, S.; Shim, K.; and Woo, J. 2004. REHIST: Relative error histogram construction algorithms. In *International Conference on Very Large Data Bases*, 300–311.

Guha, S. 2008. On the space-time of optimal, approximate and streaming algorithms for synopsis construction problems. *VLDB Journal* 17(6):1509–1535.

Halim, S., and Yap, R. H. 2007. Designing and tuning sls through animation and graphics: an extended walk-through. In *Engineering Stochastic Local Search Algorithms*, 16–30.

Halim, F.; Karras, P.; and Yap, R. H. 2009. Fast and effective histogram construction. In *International Conference on Information and Knowledge Management*, 1167–1176. New York, NY, USA: ACM Press.

Halim, S.; Yap, R. H.; and Lau, H. C. 2006. Viz: A visual analysis suite for explaining local search behavior. In *ACM Symposium on User Interface Software and Technology*, 57–66. ACM Press.

Himberg, J.; Korpiaho, K.; Mannila, H.; Tikanmäki, J.; and Toivonen, H. 2001. Time series segmentation for context recognition in mobile devices. In *IEEE International Conference on Data Mining*, 203–210. Washington, DC, USA: IEEE Computer Society.

Ioannidis, Y. E., and Poosala, V. 1995. Balancing histogram optimality and practicality for query result size estimation. In *ACM SIGMOD International Conference on Management of Data*, 233–244.

Ioannidis, Y. E., and Poosala, V. 1999. Histogram-based approximation of set-valued query-answers. In *International Conference on Very Large Data Bases*, 174–185.

Ioannidis, Y. E. 1993. Universality of serial histograms. In *International Conference on Very Large Data Bases*.

Ioannidis, Y. E. 2003. The history of histograms (abridged). In *International Conference on Very Large Data Bases*, 19–30.

Jagadish, H. V.; Koudas, N.; Muthukrishnan, S.; Poosala, V.; Sevcik, K. C.; and Suel, T. 1998. Optimal histograms with quality guarantees. In *International Conference on Very Large Data Bases*, 275–286.

Karras, P.; Sacharidis, D.; and Mamoulis, N. 2007. Exploiting duality in summarization with deterministic guarantees. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 380–389. New York, NY, USA: ACM Press.

Kooi, R. 1980. *The Optimization of Queries in Relational Databases*. Ph.D. Dissertation.

Muralikrishna, M., and DeWitt, D. J. 1988. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *ACM SIGMOD International Conference on Management of Data*, 28–36. New York, NY, USA: ACM Press.

Piatetsky-Shapiro, G., and Connell, C. 1984. Accurate estimation of the number of tuples satisfying a condition. In *ACM SIGMOD International Conference on Management of Data*, 256–276. New York, NY, USA: ACM Press.

Poosala, V., and Ioannidis, Y. E. 1997. Selectivity estimation without the attribute value independence assumption. In *International Conference on Very Large Data Bases*, 486–495.

Poosala, V.; Ioannidis, Y. E.; Haas, P. J.; and Shekita, E. J. 1996. Improved histograms for selectivity estimation of range predicates. In *ACM SIGMOD International Conference on Management of Data*, 294–305.

Salmenkivi, M.; Kere, J.; and Mannila, H. 2002. Genome segmentation using piecewise constant intensity models and reversible jump MCMC. In *European Conference on Computational Biology*, 211–218.

Terzi, E., and Tsaparas, P. 2006. Efficient algorithms for sequence segmentation. In *SIAM International Conference on Data Mining*.