NP-Complete Problems in Synthetic Disk Access Pattern Generation

Zachary Kurmas
Dept. of Computer Science
Grand Valley State University
kurmasz@gvsu.edu

1 Introduction

The performance of a disk array depends on the specific workload being used to drive the array; therefore, a useful disk array evaluation requires that the workloads be chosen carefully. Such workloads can either be traces of real, production workloads or synthetically generated (random) workloads. Both types of workloads have benefits and limitations[3, 4, 9]; therefore, some evaluations are best conducted using traces of real workloads, while other evaluations are best conducted using synthetic workloads.

To be useful, a synthetic workload must maintain a set of relevant characteristics. The particular characteristics that should be maintained vary from situation to situation; and quickly choosing an optimal set of characteristics is still an open problem. However, common characteristics include a specified percentage of reads, a distribution of request size, and a distribution of starting sectors (the first sector accessed by some request). Our research indicates that it is useful for synthetic workloads to also maintain given distributions of jump distance[1] [10, 11], and inter-reference temporal density [1]. We will show that maintaining either of these distributions while maintaining a given distribution of starting sectors is NP-Hard.

1.1 Significance

Although they have limitations, synthetic workloads play an important role in storage systems research. For example, researchers can evaluate how small changes to a workload can potentially affect the performance of a disk array by modifying the characteristics on which a synthetic workload is based. In contrast, it is usually much more difficult to deliberately modify individual characteristics of a workload trace.

Ganger demonstrated that simple characteristics often do not define a synthetic workload precisely enough accurately model the "real" workload on which it is based [4]. As a result, researchers have been developing more complex characteristics and corresponding generation techniques [8, 6, 7, 13]. Understanding the complexity of proposed generation techniques can help direct a researcher's search for an appropriate algorithm. Identifying a particular problem as NP-Hard suggests the need for an approximation algorithm. Furthermore, the problems considered for use in the proof can suggest how one could apply existing approximation algorithms to a new problem.

In particular, in the process of proving that the Jump Distance problem (discussed in Section 2) was NP-Hard, we found that it could be transformed into either a directed Rural Postman Problem, or a Traveling Salesman Problem. The literature contains several heuristics for each problem [2, 5, 12]. Similarly, by finding that the Partition problem reduces to the temporal inter-reference generation problem (discussed in Section 3), we know that a simply greedy algorithm is unlikely to produce an exact solution, and we also know where to look for an approximation algorithm.

2 Jump Distance

Generating a synthetic workload that maintains both (1) a given distribution of starting sectors and (2) a given distribution of jump distances requires that we find an ordering of the starting sectors and an ordering of the jump distances such that, for each request i, $s_i + j_i = s_{i+1}$. We will show that an exact solution to this problem is equivalent to the Hamiltonian Path Problem (HPP). In particular, we will redefine our workload generation problem as a decision problem (a problem with a yes-no answer), then we will show that the Hamiltonian Path problem reduces to the modified Jump Distance Problem (mJDP).

2.1 Problem Definition

Let R be a finite set of I/O requests; and let $0 \leq s(r) \leq K$ be the starting sector of request $r \in R$. Let J be a finite set of jumps; and let $-K \leq d(j) \leq K$ be the distance of jump

[1] For purposes of our study, we define "jump distance" as defined as the difference between the starting sector of two successive I/O requests.

j. In practice, $|J| = |R-1|$; however, for purposes of this problem, we assume only that $|J| \geq |R| - 1$. We define the modified Jump Distance Problem (mJDP) as follows: Given sets R and J as well as functions s, and d, is there an ordering of R and J such that for $1 \leq i \leq |R| - 1$, $s(r_i) + d(j_i) = s(r_{i+1})$. (Some jumps may be unused.)

2.2 Proof

We will show that the Hamiltonian Path problem (HPP) reduces to mJDP. In particular, we will show that a polynomial time algorithm for mJDP can also be used to solve HPP in polynomial time. Let $G = (V, E)$ be a connected, directed graph. Because G is connected, we know that $|E| \geq |V| - 1$. Construct an instance of mJDP as follows: For each vertex $v_i \in V$, add a request r_i to R and define $s(r_i) = 2^i$. Now, for each edge $e_i = (v_a, v_b) \in E$ add the jump j_i to J and define $d(j_i) = s(r_b) - s(r_a) = 2^b - 2^a$.

We will now (1) present a polynomial-time transformation from an instance of HPP to mJDP, and (2) show that a graph G has a Hamiltonian Path if and only if there is a sequence of requests that satisfies the corresponding mJDP.

2.2.1 Polynomial-time transformation

This transformation from HPP to mJDP can be done in polynomial time. The input size of an arbitrary graph G is $O(|E| \log |V|)$: Each of $|E|$ edges lists its stating and ending vertices (requiring $\log |V|$ bits per vertex listed). We can transform G into an instance of mJDP as follows: For each vertex $v_i \in V$, list i followed by its starting sector number 2^i. This requires $\log |V| + |V|$ bits per vertex, or $O(|V|^2)$ bits total. The running time for this portion of the transformation is also $O(|V|^2)$. For each edge e_i, list i followed by the jump distance corresponding to e_i. The description of each edge requires $\log |E| + |V|$ bits, or $O(|E||V|)$ bits for all edges. Likewise, the running time is also $O(|V||E|)$. The overall running time of the transformation is $O(|E||V|)$, which is polynomial in the size of the the graph, $O(|E| \log |V|)$, if we assume $|E| \geq |V| - 1$. (There are more efficient methods of describing and encoding mJDP, but this method more concisely demonstrates the transformation is polynomial.)

2.2.2 HPP implies mJDP solution

If a graph G contains a Hamiltonian Path, then there is a sequence of requests that satisfies the corresponding mJDP. Let $P = v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_n}$ be a Hamiltonian Path in G. We will show that $r_{i_1}, r_{i_2}, r_{i_3}, ..., r_{i_n}$ represents a valid request ordering in mJDP. Let $(v_{i_a}, v_{i_{a+1}})$ be an edge along P; and, let j_e be the unique jump corresponding to that edge. We know that $s(r_{i_a}) + d(j_e) = s(r_{i_{a+1}})$ because $s(r_{i_a})$ is defined to be 2^{i_a}, $s(r_{i_{a+1}})$ is defined to be $2^{i_{a+1}}$, and $d(j_e)$ is defined to be $2^{i_{a+1}} - 2^{i_a}$. Therefore, $r_{i_1}, r_{i_2}, r_{i_3}, ..., r_{i_n}$ represents a valid request ordering in mJDP.

2.2.3 mJDP solution implies HPP

Our last task is to show that any valid mJDP sequence corresponds to a Hamiltonian Path. We first show that for any integer $d \neq 0$, there is at most one edge $(v_1, v_2) \in E$ for which the corresponding jump has a distance of d. Let d' be the function that maps edges of G to the distance of the corresponding jump in the mJDP. From the definitions of the transformation and of the function d, we know that $d'(v_a, v_b) = 2^b - 2^a$. We now assume that $d > 0$ and show that if $d'(v_a, v_b) = d'(v_c, v_d)$ (where $a \neq b$, $c \neq d$, and a, b, c, and d are all non-negative), then $a = c$ and $b = d$. The proof where $d < 0$ is analogous. By definition of d', we know that $2^b - 2^a = 2^d - 2^c$. This is equivalent to saying $2^b + 2^c = 2^d + 2^a$. We now consider two cases:

- Assume $b \neq c$. In this case, we know that $2^b + 2^c$ is an integer whose binary representation has precisely two 1s. Because each integer has precisely one binary representation, we know that either:

  1. $b = d$ and $c = a$, or
  2. $b = a$ and $d = c$.

  Because we assume that $b \neq a$, we may conclude that $b = d$ and $c = a$ as desired.

- Now assume $b = c$. Then $2^b + 2^c = 2^b + 2^b = 2^{b+1}$. Thus, $2^{b+1} = 2^d + 2^a$. The binary representation of 2^{b+1} contains exactly one 1. Suppose to the contrary that $a \neq d$. In this case $2^d + 2^a$ would have a binary representation comprising two 1s. Therefore, we can conclude that if $b = c$, then $a = d$. From here, we can conclude that $2^{b+1} = 2^{d+1}$ and that $b = d$ and $c = a$ as desired.

Now, we show that any valid mJDP sequence corresponds to a Hamiltonian Path. Let $r_{i_1}, j_{i_1}, r_{i_2}, j_{i_2}, r_{i_3}, j_{i_3}, ..., r_{i_{n-1}}, j_{i_{n-1}}, r_{i_n}$ be a valid ordering for the mJDP. Because there is a one-to-one correspondence between elements of V and R, $r_{i_1}, r_{i_2}, r_{i_3}, ..., r_{i_{n-1}}, r_{i_n}$ defines an ordering of V. We need only show that for each $a \in [1, |V| - 1]$, $(v_{i_a}, v_{i_{a+1}}) \in E$. Because we have a valid mJDP sequence, we know that there is an edge $j \in E$ such that $d(j) = 2^{i_{a+1}} - 2^{i_a}$. The edge $(v_{i_a}, v_{i_{a+1}})$, if it exists, is a candidate for j because $s(r_{i_a}) + d(v_{i_a}, v_{i_{a+1}}) = 2^{i_a} + (2^{i_{a+1}} - 2^{i_a}) = 2^{i_{a+1}} = s(r_{i_{a+1}})$. We just showed that there is at most one edge j for which $d(j) = 2^{i_{a+1}} - 2^{i_a}$; therefore, we can we can conclude that $j = (v_{i_a}, v_{i_{a+1}})$ and $(v_{i_a}, v_{i_{a+1}}) \in E$.

Hence, we can conclude that $v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_n}$ is a Hamiltonian Path and we have shown by reduction to the Hamiltonian Path Problem, that mJDP is NP-Complete.

3 Inter-reference Temporal Density Function

The inter-reference temporal density function (IRTDF) is a measure of a cache's effectiveness on a given workload [1]. A particular request's temporal inter-reference distance is defined to be the number of unique starting[2] sectors (including the current sector[3]) accessed since the last request to the same sector. The temporal inter-reference distance of a sector accessed for the first time is defined to be 0. If a request's inter-reference distance is between 1 and the capacity of the cache, then that request will be a cache hit, otherwise it will be a cache miss (assuming a fully-associative cache with one-sector blocks and an LRU replacement policy). The IRTDF presents the distribution of inter-reference distances over all requests, thus providing an estimate of the hit rate for any given cache size. Specifying a synthetic workload's IRTDF allows the user to (approximately) specify the cache hit rate.

Generating a workload that maintains both (1) a given IRTDF and (2) a given distribution of starting sectors requires that we find an ordering of the starting sectors that maintains the given IRTDF. We will show that determining if such an ordering exists is NP-complete using a reduction from the Partition problem.

3.1 Problem Definition

Let function $l : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{Z}^{\geq 0}$ be a distribution of starting sectors. In particular, let $l(s)$ be the number of I/O requests that have a starting sector of s. (In other words, l is a density function that has been multiplied by the desired number of requests.) Let function $t : \mathbb{Z}^{\geq 0} \rightarrow \mathbb{Z}^{\geq 0}$ be a distribution of inter-reference temporal densities. In particular, let $t(i)$ be the number of requests that have an inter-reference temporal density of i. (Again, t is an IRTDF that has been multiplied by the desired number of requests.) The function l implicitly defines a set R of I/O requests. We define the temporal inter-reference generation problem (TIRGP) as follows: Does there exist an ordering σ of R such that the IRTDF of R with order σ is t?

[2] When considering an I/O workload where requests comprise more than one sector, one could also reasonably count all unique sectors, not just starting sectors.

[3] This means the temporal inter-reference distance between two consecutive accesses to the same sector is 1.

3.2 Proof

We will show that a slightly modified Partition problem reduces to TIRGP. In particular, we will show that a polynomial time algorithm for TIRGP can also be used to solve the Partition problem in polynomial time. Gary and Johnson[5] define the partition problem as follows: Given a finite set A and a size $s(a) \in \mathbb{Z}^+$, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$? We add the restriction that $\sum_{a \in A} s(a)$ is even. This modified problem is still NP-Complete and greatly simplifies the proof.

Let A be a finite set, with a size $s(a) \in Z^+$ assigned to each $a \in A$ such that $\sum_{a \in A} s(a)$ is even. Choose an arbitrary ordering for the elements of A; and, define an instance of TIRGP as follows: For each $a_i \in A$, define $l(i) = s(a_i)$. Now, define $t(0) = |A|$; and define $t(2) = |A| - \sum_{a \in A} s(a)$. All other values of t are 0. We will show that there is an ordering σ of R that satisfies t if an only if there is a solution to the Partition problem defined by A.

3.2.1 Partition Implies TIRGP

Let A' be a solution to the Partition problem defined by A. Let R be the set of requests implicitly defined by l; and, define $ssec(r)$ to be the starting sector of request $r \in R$. Define R_1 be the set of requests for which $ssec(r)$ corresponds to an element $a \in A'$. $R_1 = \{r | ssec(r) = i \text{ and } a_i \in A'\}$). Let R_2 be the set of requests for which $s(r)$ corresponds to an element in $A - A'$. Order the elements of R_1 and R_2 according to starting sectors, then define the ordering σ of R to consist of alternating requests from R_1 and R_2.

This ordering maintains distribution function l, because it is simply a permutation of requests defined by l. We now show that it also maintains the desired IRTDF t. The first access to any given sector, by definition, has a temporal inter-reference distance (IRD) of 0. These are the only references with a IRD of 0. Thus, $t(0) = |A|$: the number of unique starting sectors. We now show that each of the remaining requests has an IRD of 2.

Consider request r_i; and, assume r_i is not the first request with starting sector $ssec(r_i)$. From this assumption, we can conclude that the IRD of r_i is not 0. We can also conclude that the IRD of r_i is not 1. The only way for a request to have an IRD of 1 is for the previous request to have the same sector; however, all requests with the same starting sector are placed together into either R_1 or R_2. Because the ordering σ alternates between R_1 and R_2, it is not possible for adjacent requests to have the same starting sector. (More formally, it is not possible for $ssec(r_{i-1}) = ssec(r_i)$.) Therefore, it is not possible for any request to have an IRD of 1.

If $s(r_{i-2}) = s(r_i)$, then IRD of r_i is 2 as desired. Assume to the contrary that there is an i such that the IRD of

$r_i > 2$. This requires that $ssec(r_j) = ssec(r_i)$ for some $j < i - 2$ and that $ssec(r_{i-2}) \neq ssec(r_i)$. Such an occurrence would violate the ordering of R_1 and R_2: Either $ssec(r_j) < ssec(r_{i-2})$ while $ssec(r_{i-2}) > ssec(r_i)$ or, $ssec(r_j) > ssec(r_{i-2})$ while $ssec(r_{i-2}) < ssec(r_i)$.

Hence, we have shown that the set of requests R under ordering σ has the IRTDF t as desired.

3.2.2 TIRGP Implies Partition

Let σ be an ordering of R that has an IRTDF of t. We will show that the set of starting sectors for the odd-numbered requests define a solution to the partition problem.

Define $R_1 = \{r_i \in R | i \text{ is odd}\}$ and define $R_2 = \{r_i \in R | i \text{ is even}\}$. Define $S_1 = \{ssec(r_i) | r_i \in R_1\}$; and, define S_2 similarly. Finally, define $A' = \{a_i \in A | i \in S_1\}$.

First, we show that S_1 and S_2 are disjoint. Assume to the contrary that there exists a sector s that is an element of both S_1 and S_2. This means there exist distinct requests $r_i \in R_1$ and $r_j \in R_2$ such that $i < j$ and $ssec(r_i) = ssec(r_j)$. Let r_i and r_j be a pair for which $j - i$ is minimal. Because $t(1) = 0$, we know that $j - i \neq 1$. Because $i \neq j \bmod 2$, we know that $j - 1 \geq 3$. Because $t(1) = 0$, we can conclude $ssec(r_i) \neq ssec(r_{i+1})$ and that $ssec(r_{i+1}) \neq ssec(r_{i+2})$. We can also conclude that $ssec(r_i) \neq ssec(r_{i+2})$, otherwise r_{i+2} and r_j would contradict our choice of r_i and r_j as a pair that minimizes $j - i$. We now have that $ssec(r_i)$, $ssec(r_{i+1})$, and $ssec(r_{i+2})$ are all distinct. This means that the IRD of $r_j > 3$, which contradicts our definition of t. Thus, we can conclude that S_1, and S_2 are disjoint.

We now show that $\sum_{a \in A'} s(a) = |R_1|$. Consider an arbitrary $a_i \in A'$. The definition of our transformation specifies that $l(i) = s(a_i)$. Because S_1 and S_2 are disjoint, then each of the $s(a_i)$ requests with a starting sector of i are in R_1. Because each $a \in A'$ contributes $s(a)$ unique requests to R_1, $\sum_{a \in A'} s(a) = |R_1|$ as claimed. By similar logic, $\sum_{a \in A - A'} s(a) = |R_2|$. Because we specified that $\sum_{a \in A} s(a)$ was even, we know that $|R|$ is even. Consequently, $|R_1| = |R_2|$. As a result, we can conclude that A' is a solution to the Partition problem defined by A and also that TIRGP is NP-Complete.

4 Conclusions

We have proved that generating a synthetic workload that precisely maintains a given distribution of starting sectors along with either a distribution of jump distances or a distribution of temporal inter-reference distances is NP-Hard. By identifying these problems as HP-Hard, we know to turn our focus to approximation techniques. Furthermore, that problems we use in our proofs can serve a starting points for our search for a useful approximation algorithm.

References

[1] T. M. Conte and W. W. Hwu. Benchmark characterization for experimental system evaluation. In *Proceedings of the 1990 Hawaii International Conference on System Sciences*, volume I, pages 6–18, 1990.

[2] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part ii: The rural postman problem. *Operations Research*, 43(3):399–414, 1995.

[3] D. Ferrari. *Computer Systems Performance Evaluation*. Prentice-Hall, Inc., 1978.

[4] G. R. Ganger. Generating representative synthetic workloads: An unsolved problem. In *Proceedings of the Computer Measurement Group Conference*, pages 1263–1269, December 1995.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[6] M. Gomez and V. Santonja. Self-similarity in I/O workload: Analysis and modeling. In *Workshop on Workload Characterization*, November 1998. Workshop held in conjunction with the 31st annual ACM/IEEE International Symposium on Microarchitecture, Dallas, Texas.

[7] M. E. Gomez and V. Santonja. A new approach in the analysis and modeling of disk access patterns. In *Performance Analysis of Systems and Software (ISPASS 2000)*, pages 172–177. IEEE, April 2000.

[8] B. Hong, T. Madhyastha, and B. Zhang. Cluster-based input/output trace synthesis. Technical report, University of California at Santa Cruz, 2002.

[9] Z. Kurmas, K. Keeton, and K. Mackenzie. Iterative distillation of I/O workloads. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2003.

[10] Z. Kurmas, J. Zito, L. Trevino, and R. Lush. Generating a jump distance based synthetic disk access pattern. In *14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies*, May 2006.

[11] Z. Kurmas, J. Zito, L. Trevino, and R. Lush. Generating a jump distance based synthetic disk access pattern. Technical Report GVSU-CIS-2006-01, Grand Valley State University, 2006.

[12] Lin, S. and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, mar 1973.

[13] M. Wang, T. M. Madhyastha, N. H. Chan, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *Proceedings of the 16th International Conference on Data Engineering (ICDE02)*, 2002.