

Thomas W. Tiahart

Department of Computer Science
Northwestern College
101 7th St. SW
Orange City, IA 51041

Phone: (712) 707-7052
Email: thomas.tiahart@nwciowa.edu

Education

Ph.D., Computational Science and Statistics, in progress (expected Fall 2010),
University of South Dakota.

M.A., Computer Science, University of South Dakota, 1991.

B.S., Accounting, University of South Dakota, 1982.

Research

Ph.D. Dissertation: *Non-comparison Set Intersection Trees* – This non-comparison approach to the solution of the set intersection problem replaces individual element selection with bitwise elimination of either elements or subtrees, to reduce search complexity below the information theoretic limit imposed by the comparison model.

VisualMetrics Technical Report: *Micro Search Analysis* – Literature survey and analysis of the state of the art of text processing in information retrieval as of March 1997.

Master's Thesis: *A Decision Support Forecasting Tool for a Temporal Database* – Integrated three forecasting models into a simulated temporal database.

Teaching Experience

1988–1991 *Microcomputer Applications* – Graduate Teaching Assistant in Computer Science – taught operating system, word processing, spreadsheet and database management use.

Teaching Philosophy

I find that I learn best by acquiring knowledge incrementally, in the strata above the trivial and below the inaccessible. I aspire to find the stratum for each student that is challenging, but also provides a path for success.

Students have different learning modes. Some learn by seeing theory and principles first, then applying them to problems. For others, examples reveal the principles. Still others must write documents, code or draw diagrams to understand. One attractive quality of computer science is that there are many opportunities for active learning. I believe all of these paths can be incorporated into the curriculum.

Teaching Objective

Both my education and experience have taught me that computer science is more than software development. Computer science is a disciplined way of thinking, enabling practitioners to traverse multiple levels of abstraction. Complexity can be managed only through abstraction. But the ability to manipulate abstractions is applicable to more than computer science. In systems at scale, people, along with their objectives, values,

and desires interact with physical, organizational and electronic infrastructure, to produce complex relationships. Computer scientists are well equipped to develop systems and products that enable organizations to attain their goals.

My objective in teaching is to equip students to think systematically, consistently, and computationally, then realize the resulting abstractions in concrete implementations. Success in this means success not only in software development, but in other aspects of life as well.

Employment

2010–Instructor in Computer Science, Northwestern College

2008–2010 Computational Science Research Assistant, University of South Dakota

2006–2008 Chief Scientist, BrightPlanet Corporation

2003–2006 Principal Software Engineer, BrightPlanet Corporation

1996–2003 Principal Software Engineer, VisualMetrics Corporation

1995–1996 Senior Software Engineer, Gateway

1993–1995 Senior Software Engineer, Parallel Software, Inc.

1991–1993 Senior Design Engineer, Mycro-Tek, Inc.

1988–1991 Graduate Teaching Assistant, University of South Dakota

1986–1988 Director of Software Development, CMC Information Systems, Inc.

1983–1986 Software Engineer, CMC Information Systems, Inc.

Patents

US Patents 7,676,555, 7,146,409; *System and method for efficient control and capture of dynamic database content*, March 9, 2010. Inventors: Bushee; William J., **Tiahart; Thomas W.**, Bergman; Michael K.

US Patent 7,249,122; *Method and system for automatic harvesting and qualification of dynamic database content*, July 24, 2007. Inventors: Bushee; William J., **Tiahart; Thomas W.**, Bergman; Michael K.

US Patent 6,741,979; *System and method for flexible indexing of document content*, May 25, 2004. Inventors: **Tiahart; Thomas W**

Selected Software Development Experience

Chief Scientist, BrightPlanet Corporation.

Researched, developed and had overall responsibility for the Information Retrieval (IR) software that comprises BrightPlanet's indexing subsystem. Designed and wrote the SQSTR (Semantic Query, Storage, Transformation, and Reporting) subsystem, adding XML processing to existing IR capabilities. Also rewrote the performance bottleneck portions of the distributed processing middleware, including results consolidation and the native interface.

Principal Software Engineer, BrightPlanet Corporation.

Researched, analyzed, developed, tested and deployed the document-centric BrightPlanet Text Engine. The Text Engine is an inverted index IR system that employs a trie-based dictionary, full-text postings with positions, and two-phase document compression. Reduced document size through token-to-integer conversion, and Huffman encoding of converted integers. The Text Engine also implemented index-only query processing using standard Boolean operations, positional/proximity operators, phrases, wild cards and regular expressions. Together with filter lists, the text engines query processing subsystem provides on-demand dynamic results for the hierarchical categorization system, without the need for manual document tagging.

Principal Software Engineer, VisualMetrics Corporation.

Researched, analyzed, developed, tested and deployed the Mata Hari/Lexibot meta-search tools inverted index text processing subsystem. This index was used initially in a desktop application to search multiple search engines simultaneously. Later, the same underlying technology was adapted to a query and rule-based static categorization system. The categorization system used inheritance-derived inclusion and exclusion lists in conjunction with queries to place documents in a hierarchical node structure.

Senior Software Engineer, Gateway.

Created an order suspension and release subsystem supporting multiple departments and incorporated into two separately developed applications with any interface or underlying code changes.

Senior Software Engineer, Parallel Software.

Worked under contract to Loral Medical Imaging Systems. Conducted requirements analysis, coded, tested the Picture Archive Communication Systems (PACS) Removable Media subsystem. This module allowed physicians to record images and annotations on local media that could be carried with them between offices and hospitals.

Senior Software Engineer, Parallel Software.

Contracted with to Siemens Gammasonics for software development. Identified, described and documented the image loss potential during PACS network or server failure. Coded and tested the Failover Image Recovery system, which stored images locally during failover. This saved images until they could be uploaded once PACS was restored to a fully-operational status.

Director of Software Development, CMC Information Systems

Managed the software development staff, estimated and managed projects, constructed budgets, tracked expenses, hired and fired staff.

Software Engineer, CMC Information Systems

As a software engineer, met with customers to develop requirements; designed, constructed, installed and modified accounting and database applications for small to medium sized businesses. Replaced hard-coded procedures necessary to install basic software packages with a parameter-based installer, and implemented code-reuse procedures, libraries and tools that simplified the customization of common financial packages.

Notable Software Development Achievements

BrightPlanet Corporation

Rewrote the query-results consolidation routines for the distributed processing aggregator. Replaced the sort among all results of all nodes ($O(n \log n)$) with a linked list by score. This reduced processing time to $O(n)$, which was both faster and simpler than the so-called Ideal Merge ($O(n \log m)$). Used one singly-linked list for each score, and round-robin processing to place results into output for optimal distribution.

Constructed a regular expression processor by implementing a deterministic finite automata (DFA) built from a parse tree. The DFA was evaluated against the index, not the original document, reducing run times.

UTF-8 Case Folding – wrote a code generation module to convert a UTF specification into in a C code case-folding routine. Compiled and linked the output to produce a DFA. This included validity enforcement, as well as handling UTF-8 multiple byte codes as a single unit.

Eliminated the sort of triples during index construction process (cf. *Managing Gigabytes*, Witten, Moffat, Bell) through the use of direct addressing, Virtual Memory and a singly linked list. This reduced build time for our largest dataset from 14 hrs 35 minutes to 1 hr 45 minutes.

Replaced individual terms with stem ordinal sets in query processing. All higher level query processing remained the same, only the lowest-level code required modification. This allowed regular expression sets, including wildcards, to behave in exactly the same way as single terms.

Substituted a sort of document ordinals and document positions from multiple terms or stem ordinal sets (which are already in ascending order), with a priority queue. This eliminated 90% of the run time in the test set for query evaluation and scoring.

Improved query evaluation execution times by identifying and processing the smallest component of a Boolean expression to either include or exclude documents.

Implemented a document similarity operation allowing a new document to be used to retrieve the most similar documents from the collection. This supports identification of similar funding requests to prevent duplication and to ensured relevant preexisting work was included in further research.

Constructed a document compare routine based on integer-valued word ordinals instead of text terms, which eliminated 85% of the comparison execution time.

Produced categorization run-time reductions by substituting bit vectors for Boolean queries. The categorization taxonomy was hierarchical with inheritance. Because the operations were Boolean OR, and Boolean AND NOT, inclusion of documents with terms and exclusion of documents with terms could be performed using bit vectors. This reduced a 24 hour categorization run time to 30 minutes.

In replacing the internal paging system with one based on memory-mapped files and virtual memory I reduced index retrieval time by 30%.

Sped free list access by maintaining 16 free lists for each 4 GB segment, and accessing free list requests by promoting requests to the next higher free list. Requests therefore were guaranteed to be met on the first free-list probe instead of traversing the list to locate a suitable block.

Precomputed the square roots from 1-10000 to speed Extended Boolean Information Retrieval computation. Compared to the Standard C Library floating point routine this reduced the computation time to one-third of the original.

Gateway

Created a three-tier order suspension and release subsystem using Microsoft Foundation Classes (MFC), C++ and Sybase. This subsystem supported order suspension for the international, tax, finance, sales, personnel and accounts receivable departments. Its user-interface was developed with MFC, and incorporated a decision object layer as well as a database interface layer. The latter directly accessed a Sybase database for data

storage and retrieval. Once complete, two separate and independently developed applications, one 16-bit, one 32-bit, integrated this subsystem without modification.

Parallel Software, Inc.

Developed software for the Siemens/Loral PACS software suite, a Picture Archiving Communication System. PACS automates all medical image handling for large hospitals, or groups of hospitals.

Enhanced and extended both the Computed Radiography control module for a Fuji plate reader and the Video Acquisition Workstation (VAW) application. Solely created the Failover Image Recovery (FIR) application, which transfers images initially saved on a local Macintosh workstation during network or server failure to the server after failure resolution. Along with writing new code, this required extraction and modularization of large existing code segments from two other applications. Prior to FIR completion, local images could only be viewed on the original acquisition workstation. Developed a removable media application which supported medical image storage on local Macintosh removable media drives.

Memberships and Honors

University of South Dakota John W. Carlson Research Grant

Upsilon Pi Epsilon, International Honor Society for the Computing and Information Disciplines

Omicron Delta Epsilon, the International Honor Society for Economics

The Association of Computing Machinery, ACM SIGIR, ACM SIGMOD

University of South Dakota Outstanding Computer Science Graduate Student 1990

University of South Dakota Outstanding Computer Science Teaching Assistant 1991

Mycro-Tek Employee of the Quarter, Fourth Quarter, 1992