"Intelligent Heuristics Are the Future of Computing"

SHANG-HUA TENG, University of Southern California (USC), USA

Back in 1988, the partial game trees explored by computer chess programs were among the largest search structures in real-world computing. Because the game tree is too large to be fully evaluated, chess programs must make heuristic strategic decisions based on partial information, making it an illustrative subject for teaching AI search. In one of his lectures that year on AI search for games and puzzles, Professor Hans Berliner—a pioneer of computer chess programs[1]—stated:

"*Intelligent heuristics are the future of computing.*"

As a student in the field of the theory of computation, I was naturally perplexed but fascinated by this perspective. I had been trained to believe that "Algorithms and computational complexity theory are the foundation of computer science." However, as it happens, my attempts to understand heuristics in computing have subsequently played a significant role in my career as a theoretical computer scientist. I have come to realize that Berliner's postulation is a far-reaching worldview, particularly in the age of big, rich, complex, and multifaceted data and models, when computing has ubiquitous interactions with science, engineering, humanity, and society. In this article,[2] I will share some of my experiences on the subject of heuristics in computing, presenting examples of theoretical attempts to understand the behavior of heuristics on real data, as well as efforts to design practical heuristics with desirable theoretical characterizations. My hope is that these theoretical insights from past heuristics—such as spectral partitioning, multilevel methods, evolutionary algorithms, and simplex methods—can shed light on and further inspire a deeper understanding of the current and future techniques in AI and data mining.

CCS Concepts: • **Computing methodologies**; • **Theory of computation** → **Theory and algorithms for application domains**; **Design and analysis of algorithms**; **Computational complexity and cryptography**; • **Mathematics of computing** → **Stochastic processes**; • **Applied computing** → **Operations research**;

Additional Key Words and Phrases: Heuristics in computing, data mining, AI, network analysis, data analysis, deep learning, spectral graph theory, multilevel methods, smoothed analysis, beyond worst-cast analysis, axiomatic approach, linear programming, evolutionary algorithm, local clustering, robust statistics, game trees, binary decision diagram, PageRank, spectral graph sparsification, dimensionality reduction, Shapley value, network influence, network centrality

[1]Professor Berliner created the AI-based HiTech, the first computer chess program to reach the senior master (2400) on the United States Chess Federation rating.
[2]This article is based in part on my keynote talk at KDD'22 discussing my manuscript, "Scalable Algorithms for Data and Network Analysis" [135]. The talk itself can be found at https://www.youtube.com/watch?v=mkZ12sRpK5I. Although the printed title is "Beyond Traditional Characterizations in the Age of Data: Big Models, Scalable Algorithms, and Meaningful Solutions," the talk has an informal title, "Beyond Theoreticians' Struggles with Heuristics."

This article has been supported in part by a Simons Investigator Award from the Simons Foundation.
Author's address: S.-H. Teng, University of Southern California (USC), 941 Bloom Walk, Los Angeles, California, USA, 90089; e-mail: shanghua@usc.edu.

2157-6904/2023/11-ART96
https://doi.org/10.1145/3627708

ACM Reference format:
Shang-Hua Teng. 2023. “Intelligent Heuristics Are the Future of Computing”. *ACM Trans. Intell. Syst. Technol.* 14, 6, Article 96 (November 2023), 39 pages.
https://doi.org/10.1145/3627708

1 INTRODUCTION

We live in a time of great change in computing, and the change has been fast. Back in 1988, when I was a Ph.D. student, we were taught that the field of computing could be broadly divided into four major areas: ***Artificial Intelligence*** **(AI)**, *Hardware Systems*, *Programming Systems*, and *Theory of Computing*. In fact, in the 1980s, Ph.D. students of the **Carnegie Mellon University (CMU)** Computer Science Department[3]—with about 30 new students a year—were only required to take and pass four core courses, colloquially known as the AI Qual, Hardware Qual, PS Qual, and Theory Qual.[4] All other courses were optional. However, every Ph.D. student was required to conduct research in at least two of these four areas: one as the major focus (including the Ph.D. thesis) and one as a minor.

Over the next three and a half decades, the universe of computing has rapidly expanded, with interconnected galaxies represented by premier conferences such as KDD, SIGGRAPH, and NeurIPS. Advances in computing have been driven not only by progress in data, communication, and computing infrastructure, but perhaps more significantly by the pervasive integration of computing technologies across all other academic disciplines and throughout society. These interactions have, in turn, introduced *rich, complex, multifaceted* data and models into computing, whose scale and complexity continue to challenge today’s computing power and algorithmic feasibility.

While the computer science program at CMU in the 1980s may seem comparatively narrow and specialized by today’s standards, each of the four core courses covered topics with components that have had significant and far-reaching implications on present technologies. For example, the Hardware Qual taught us how instrumental parallelism would be in future high-performance computing systems, and that parallelism exists in multiple forms, ranging from concurrency in multiprocessors to pipeline in circuits and systolic arrays. The PS Qual taught us about modern operating systems, the significance of strongly typed programming languages, and the importance of code reusability in building large and complex software systems. The Theory Qual taught us about the asymptotic framework for complexity theory and algorithm analysis, on-line decision making, and the profound implications of cryptography in secure multi-party computation.

Being the subject furthest away from my previous research and coursework,[5] the AI Qual was the most challenging course for me. That 1988 AI Qual was led by Professors Hans Berliner, Tom Mitchell, and Jaime Carbonell, with lectures given by the entire CMU AI group.[6] Professor Berliner started the course by introducing game trees associated with board games like Chess and GO, and decision trees for puzzles like Rubik’s Cube.

[3]Later in that semester, in December 1988, CMU elevated the Department of Computer Science to School of Computer Science, with then Chair Nico Habermann as its first Dean. The announcement outside the Chair’s office in Wean Hall was an iconic moment and memorable for many of us.
[4]”Qual” stands for “Qualifiers” for brevity.
[5]Before CMU, I was a Ph.D. student at USC for three years. In my first year there, under the guidance of Professor Kai Hwang, my research focused on computer architecture and parallel processing. A year later in 1986, influenced by Professors Gary Miller and Len Adleman, I switched to theoretical computer science. In 1988, I followed my Ph.D. advisor Professor Miller to CMU.
[6]The AI group at the time included Professors Herbert Simon, Allen Newell, Raj Reddy, Takeo Kanade, Paul Rosenbloom among others. Professor Geoffrey Hinton left a year before in 1987 to the University of Toronto, but his Boltzmann machines remained one of the subjects for the AI Qual.

As mathematical models, game/decision trees attracted my attention. Studying algorithms and complexity, I was drawn to natural models whose size could grow exponentially in the size of the rules that define them. Nowadays, these models are known as *big models with succinct representations.*[7] Berliner's discussion of game tree search resonated with me. When I was a Ph.D. student, I spent a lot of time reading research papers in university libraries. In addition to planned readings (proceedings/technical reports/journals/books), each time I would borrow a few issues of *Scientific American.* I always enjoyed the mathematical puzzles and games in that storied magazine. *Recreational mathematics* was effective in breaking up long research readings into attainable segments.

Professor Berliner presented various strategies and algorithms for game tree search: depth-first search, breadth-first search, alpha-beta pruning, A^* search, B^* search, and so on. In explaining his insights on the comparative strengths and weaknesses of these AI search methods, he stated:

"*Intelligent heuristics are the future of computing.*"

This sentiment was subsequently echoed throughout the AI Qual by different professors discussing various AI topics.

As a theory student, I was trained to believe that:

"*Algorithms and computational complexity theory are the foundation of computing.*"

Regarding board games, I had learned from my field of research that many of them—notably generalized Chess and GO—are PSPACE-hard to solve [64, 102, 114]. In other words, under the complexity-theoretical assumption P ≠ PSPACE, there is no efficient, i.e., polynomial time, general-purpose technique to effectively evaluate the exponential-sized game trees. Thus, I wondered that if no polynomial-time efficient solution existed, then how could one state that some search methods were more "intelligent" than others? How should one characterize such a seemingly paradoxical definition? I was puzzled but deeply fascinated by Berliner's perspective. Conceptually, I also began to struggle in this course.[8]

But, as it turned out, my attempts at understanding heuristics in computing have played a major role in my career. Today, when computing has ubiquitous interactions with every other discipline, the computational problems we face are much more complex in nature and massive in scale. Many of them cannot be solved efficiently or effectively according to traditional complexity-theoretical frameworks that are based on worst-case analysis. Heuristic decisions are required when faced with big data, just like what computer chess programs must do in evaluating the gigantic game tree. Such decisions are also crucial in the real-time formulation of complex, high dimensional data-driven models. "Intelligent heuristics are the future of computing" has indeed become a far-reaching worldview, as we have entered the age of big, rich, complex, and multifaceted data and models.

In the subsequent discussions, we will review various analyses aimed at understanding heuristics in computing, as well as explore attempts to design practical heuristics with desirable theoretical characterizations. After presenting three short stories in Section 2 that highlight basic heuristic methods—binary decision diagrams, spectral graph partitioning, and the multilevel

[7]We will have more examples in Section 5.

[8]In AI Qual, I earned the only *remedial* grade in my career as a student. In the first take-home exam for the AI qualifier, when asked to give an AI solution to a large Rubik's Cube, I restated the result of Furst, Hopcroft, and Luks [69] that these types of group-theoretical problems can be solved in polynomial time. I got zero points for that answer, which eventually landed me in remedial status. I also struggled to follow the lecture on Boltzmann machines. In the end, Professor Berliner kindly released me from the remedial after he discussed the case with Professor Merrick Furst, who was at the time also a faculty member of CMU, about his polynomial-time algorithms for permutation groups.

method—and their practical applications, we organize our main discussions in three parts: (i) understanding heuristics, (ii) designing heuristics and formulating heuristic theories, and (iii) characterizing heuristic solutions. We then draw connections between these classical heuristics and the emerging methodologies for Big Data. I believe that by reflecting on the theoretical foundations of past heuristics, we can gain invaluable insights that enrich our understanding of current and upcoming AI and data-mining methodologies. The mathematical structures underlying earlier computational challenges, which prompted the use of heuristics, combined with the theoretical characterizations of their practical solvability, can provide examples to help motivate the next generation of theories for developing and analyzing intelligent heuristics.

2 HEURISTICS IN COMPUTING: LEARNING AN ABSTRACT CONCEPT FROM CONCRETE EXAMPLES

"What are heuristics?" I asked Professor Berliner after that class. As English is my second language, the word "heuristic" was somewhat unfamiliar to me.

His reply, though somewhat indirect, pertained to large-scale search; he explained that heuristics are methods used to make intelligent decisions with incomplete information, a necessary approach when faced with an extremely large search space.

"Like in on-line decision making?" I asked.

"Not exactly." He stated and went on to explain that in AI, the extensive search space and the lack of structure enabling significant reduction—such as typically found in binary search—pose fundamental challenges to search algorithms.

Still puzzled, I posed the same question to Feng-Hsiung Hsu, a fellow Ph.D. student and a member of the CMU "Deep Thought" team.[9]

"Heuristics are approaches that work well in practice." His answer was short and concise.

On the way back to my office, I stopped by to check in with a fellow theory student, David Applegate. David loved mathematical games. Part of his research at the time was a computer analysis of Sprouts [12].

"What are heuristics?" I asked him earnestly.

"Heuristics are intuitive and effective methods without theoretical guarantees," David replied after some deliberation.

During my next meeting with my advisor, Professor Gary Miller, I stated the following.

"I am lucky that I study algorithms, not heuristics."

"Hmm, come to think of it, randomized algorithms are somewhat heuristic." Miller's response surprised me.

While these answers remained too abstract for me, they expanded my understanding of algorithmic methods, particularly for those used in practice. The fuzzy concept of a heuristic began to emerge in my mind, and I started noticing its occurrences in various instances. Over the

[9]Feng-Hsiung went on to design the IBM Deep Blue chess computer that defeated the world chess champion Garry Kasparov in 1997.

following decade, three "destined exposures" provided me with illuminating examples of heuristics in computing.

2.1 Binary Decision Diagrams

My eyes were opened in my second year at CMU by a talk of Professor Randy Bryant that I attended. The subject was ordered binary decision diagrams for Boolean functions [43]. Instead of a binary decision tree for a Boolean function, a ***Binary Decision Diagram*** **(BDD)** uses a **directed acyclic graph (DAG)** to further reduce the size of the representation by repeatedly combining equivalent structures (namely, isomorphic substructures) [42].

In an ***ordered*** **BDD (OBDD)**, introduced by Professor Bryant, all branches of binary decisions must follow the same variable ordering [42, 43]. The uniform ordering of variables across decision branches is instrumental in supporting some basic operations in various tasks for logic synthesis and model checking. Given a Boolean function f and a variable ordering π, Professor Bryant provided efficient algorithms for constructing the $\text{OBDD}_{f,\pi}$ with complexity polynomial in the output size. In the talk, he illustrated that the ordering π could have a significant impact on the size of the $\text{OBDD}_{f,\pi}$, and highlighted the importance of ordering optimization in the design of BDDs. He also demonstrated with examples that OBDDs were usually significantly more compact than traditional *normal forms* for Boolean functions.

The binary decision trees for Boolean functions reminded me of the game trees from the AI Qual. Just like how good players would recognize identical game positions regardless of how they were reached during game play, reducing the size of logic representations by merging isomorphic logic substructures sounded natural to me.

Shortly after that talk, Bryant's student Manpreet Khaira,[10] who brought me to the talk, began to work with my advisor on a survey paper on optimal orderings for Gaussian elimination [105]. He hoped to draw insights from this classical ordering problem to improve solutions for ordering optimization in BDD design. In our office, Manpreet often discussed both ordering optimization problems with me. So, I got the opportunity to appreciate the subject of OBDDs at close range. Inspired by efforts in algorithm design that used the concept of treewidth to characterize polynomial-time solvability for NP-hard graph problems [31], I was even able to formulate a research question: "Suppose one is assured that the underlying logic function f has a polynomial sized OBDD, but is not provided with any ordering for realizing such an OBDD. Then, can some NP-hard logic design tasks, such as in synthesis or layout, be solved in polynomial time?"

Bryant's work provided me with a concrete example of heuristics in computing: On the one hand, his OBDD framework provides a powerful tool for practical logic design and formal verification. On the other hand, from the traditional theoretical perspective, his framework did not produce a polynomial data structure for Boolean functions. Nevertheless, his framework struck me as principled when confronted with intractable tasks characterized by exponential lower bounds on the size of OBDDs for certain natural Boolean functions and the NP-hardness of the ordering optimization problem [42, 43].

2.2 Spectral Graph Partitioning

Nowadays, spectral methods are popular tools for data mining, image processing, network analysis, and machine learning. These methods arose in the 1970s [47, 59, 60, 67, 68, 81], and became widely used in parallel scientific computing and circuit design in the early 1990s [5, 111].

[10]Manpreet Khaira was my officemate during my first two years at CMU. A President's Gold Medalist of IITs, he came to CMU in 1988, and became Professor Randy Bryant's student. Passionate and persuasive, Manpreet convinced me to attend various talks/lectures in his field.

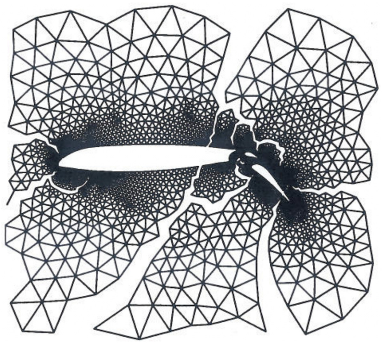

Fig. 1. Multiway geometric partitioning of NASA's airfoil finite-element mesh.

A summer visit to NASA in 1993 introduced me to this new research area.[11] Upon entering the impressive Moffett Field in Mountain View, I learned that NASA used the spectral partitioning code developed by Horst Simon's group in their parallel scientific simulation software. Simon invited me to NASA in part, because he learned from John Gilbert and Rob Schreiber that I implemented a mesh partition method with a mathematical guarantee on the partition quality in three dimensions (Refer to Figure 1 for a 2D illustration on the airfoil finite element mesh generated by Tim Barth at NASA Ames Research Center).

The NASA team planned to run the in-house spectral partitioning code simultaneously with the new geometric mesh partitioning code for their task to achieve load balancing in large-scale meshes for parallel fluid-dynamic simulations. This way, the latter could cross-validate the former. Thus, the latter's theoretical guarantee could be transferred in practice to the more widely used spectral partitioning code.

Mathematically, the geometric mesh partitioning code utilizes more data from a mesh: It takes input of the form $M = (G, xyz)$, where $G = (V, E)$ defines the graph/topological structure of the mesh and xyz provides its geometric data (that is, coordinates of the nodes). In contrast, the spectral partitioning code only uses the combinatorial data $G = (V, E)$. It first computes the eigenvector $\mathbf{v}_2$ associated with the second smallest eigenvalue λ_2 of the Laplacian matrix[12] $L_G = D_G - A_G$ (or of the normalized Laplacian matrix $\mathcal{L}_G = I - D_G^{-1/2} A_G D_G^{-1/2}$), where D_G is the diagonal matrix with node degrees and A_G is the adjacency matrix. It then uses $\mathbf{v}_2$ to "embed" G in a one-dimensional geometric space in the process of computing a partition $(S, V - S)$.

Three quantities were used to measure the quality of a partition:

$$\text{cut}(S, V - S) \quad = \quad |\{(u, v) \in E \quad | \quad (u \in S) \wedge (v \notin S)\}|, \tag{1}$$

[11]During my first year teaching at the MIT mathematics department, I received a special invitation from Dr. Horst Simon of NASA Ames Research Center. The fluid dynamic group in his division planned to incorporate the geometric mesh partitioning codes that I developed with my Xerox PARC postdoc mentor John Gilbert into their parallel simulation software [71]. In my Ph.D. thesis, "Points, Spheres, Separators: A Unified Geometric Approach to Graph Partitioning" [132], I designed a linear-time algorithm for partitioning nearest-neighbor graphs (k-NNGs) and well-shaped meshes in any fixed dimension. This theoretical algorithm crucially uses the geometric data of the input graph to obtain a provably good partition. At Xerox PARC, Gilbert suggested that we implement our first version in MATLAB, using the sparse matrix package that he and Robert Schreiber—of NASA Research Institute for Advanced Computer Science (RIACS)—had just developed for MathWorks.

[12]In the field, the λ_2 and $\mathbf{v}_2$ are called the *Fiedler value* and *Fiedler vector* of G, respectively.

$$\text{cut-ratio}(S, V - S) = \frac{\text{cut}(S, V - S)}{\min(|S|, |V - S|)}, \tag{2}$$

$$\text{conductance}(S, V - S) = \frac{\text{cut}(S, V - S)}{\min(\text{vol}(S), \text{vol}(V - S))}, \tag{3}$$

where vol(S) denotes the total node degree over nodes in S.

In a month, we converted the geometric mesh partitioning code from MATLAB to Fortran. Remarkably, on nearly all meshes we tested, the quality of each partitioning method, as measured by cut-ratio, was within five percent of the other. However, visually, the spectral method seemed to find more elegant partitions than the geometric method. We were delighted by the experimental results:

> *A theory that worked in practice and a practical solution that matched the theoretical design.*

"Like I told you earlier, all these experiments at Intel, Boeing, Thinking Machine, Cray, and here have indicated that spectral partitioning consistently works well for these meshes. There must be a theory to explain all this." Simon said at the end of a meeting.

"And you guys have the right background to find that theory," he further encouraged me.

2.3 Multilevel Methods

In the late 1970s, multilevel methods were developed to solve boundary value problems [37]. To compute a numerical solution to a **partial differential equation (PDE)**, this family of hierarchical methods uses a series of nested regular grids (or meshes) $G_0, \ldots, G_L$ to define a sequence of *finite-difference* linear systems $A_i x_i = b_i$. At the finest level, the system $A_0 x_0 = b_0$ is the traditional equation for the boundary value problem. Other smaller linear systems provide hierarchical support to solve this equation. For example, one can first solve the coarsest system $A_L x_L = b_L$, and then move the solution vector up and down the hierarchy to iteratively improve its precision.

Mathematically, grids at different levels help correct numerical errors associated with different spectral frequencies in the approximate solutions. Thus, it is beneficial to move the solution vector up and down the multilevel structure to achieve the following goals: (1) using coarser grids to efficiently obtain better initial solutions for the finer grids, and (2) using an iterative algorithm at all levels to obtain more precise approximations by reducing numerical errors at various spectral frequencies. Careful interactions between *local* iterative improvements and *hierarchical* transformations of the solution vectors have led to fast and accurate numerical solutions, as exemplified by the multigrid methods [36, 38].

My first introduction to multilevel methods was in the summer of 1994, when I was invited to give a talk titled, "A geometric approach to parallel hierarchical and adaptive computing on unstructured meshes," at the *SIAM Conference on Applied Linear Algebra* in Snowbird, Utah. After my talk, Robert Schreiber invited me to join his dinner group, and I happened to sit next to Gene Golub.[13] Professor Golub commented on the similarity between the title of my talk and that of Brandt's pioneering paper "Multilevel adaptive solutions to boundary value problems." He told me that more theoretical work was needed to extend the multigrid analyses from regular grids

[13]Gene Golub was among the foremost pioneers of numerical analysis and scientific computing. His book "*Matrix Computation*" co-authored with Charles Van Loan [72] very influential in the field and has recently found wider applications in data mining, network analysis, and AI, especially for designing algorithms using singular value decomposition (SVD), eigenvalues, and eigenvectors.

to unstructured meshes. He further encouraged me to build connections between my work and multilevel methods.

That September, I started my tenure-track position at the University of Minnesota.[14] A group of new colleagues helped to broaden and deepen my interests in this family of heuristics. Vipin Kumar and George Karypis showed me their ongoing METIS project [90]. They told me that multilevel methods had already emerged as a powerful technique for combinatorial optimization, particularly for partitioning. For example, Barnard and Simon [20] demonstrated with experiments that their multilevel code was faster for computing an approximate Fiedler vector than direct applications of the traditional Lanczos algorithm. Bui and Jones [44] used the following multilevel method for graph partitioning:

(1) Build a sequence of graphs $G_0, \ldots, G_L$ as follows: Let $G_0 := G$, and for $i = 1$ to L, find a random maximal matching in G_{i-1} and obtain a coarser graph G_i from G_{i-1} by contracting the matching edges. Then find a good partition for G_L.
(2) From coarsest to finest, project the partition from G_{i+1} to G_i, and apply the Kernighan-Lin local search heuristic [94] to improve the partition of G_i. Return the partition at G_0.

Kumar and Karypis' multilevel METIS—an improvement upon the CHACO package developed by Hendrickson and Leland [83] at the Sandia National Laboratories—became the gold standard for graph partitioning. Both CHACO and METIS refined the matching-contraction process for graph coarsening. While CHACO used spectral bisection at the coarsest level, METIS used a more sophisticated partitioning method for the coarsest level, making it more natural to extend to multi-way and hypergraph partitioning [91]. These heuristics may appear *ad hoc* to theoreticians, but their reputation was *real*. I was deeply impressed by the remarkable experimental results of these multilevel partitioning codes on various graphs that arose in applications.

"The field would benefit from further theoretical analysis of these multilevel methods to enhance our understanding them." Like Simon, Kumar and Karypis encouraged me to think about mathematical guarantees of these practical methods. Although I did not make any progress on Kumar-Karypis' question during my three years at Minnesota, Golub's question about unstructured meshes motivated my work, "Optimal good-aspect-ratio coarsening for unstructured meshes" with Miller and Talmor [106]. When I showed the paper to Golub while visiting Stanford in 1997, he smiled and said: "I am still waiting for your paper on the multilevel methods."

Two years later, I wrote a survey paper, "Coarsening, Sampling, and Smoothing: Elements of the Multilevel Method" [133]. It was my first paper not about the theory of algorithm design and analysis, but about basic techniques underlying a family of practical heuristics. Five years later, Spielman and I applied the multilevel framework to develop the first scalable algorithm for solving Laplacian linear systems [120].

3 UNDERSTANDING HEURISTICS: BEYOND WORST-CASE ANALYSIS

"There must be a theory to explain all this." Horst Simon's sentiment regarding spectral partitioning codes on real data (in the field of scientific computing and VLSI design) extended to many other wonderful heuristics in computing.

Back then, the traditional theory of computing usually characterized an algorithm's performance—time complexity, space complexity, or solution optimality—according to its

[14] As the home of the NSF Geometry Center, NSF Institute for Mathematics and its Applications (IMA), Army High Performance Computing Research Center, and being within driving distance to the Cray Computer Corporation, the university had a strong computational science program.

asymptotic worst-case measures. The *asymptotic* perspective of algorithms is now pertinent to practical computing, particularly as the sizes of problems—like in today's mining and learning tasks over big data—become massive. However,

> *The worst-case performance measurement has continued to be the source of the gap between theory and practice.*

Having a good worst-case performance is an exceptionally high requirement and guarantee in algorithm design. It is sometimes necessary in fields such as cryptography and security, but:

> *"It is commonly believed that practical inputs are usually more favorable than worst-case instances." [123]*

Understanding heuristics across vast computational applications requires theories and mathematical analyses going beyond worst-case to capture the essence of performance on real data.

3.1 Spectral Partitioning Works: Condition-based Analysis

The spectral partitioning codes, as used by NASA, had several options for generating a partition from the Fiedler vector: `sign`, `bisection`, `gap`, and `ratio`. Suppose $\mathbf{v}_2 \in \mathbb{R}^{|V|}$ is the Fiedler vector of an input graph $G = (V, E)$. Then, for each *splitting value* $s \in \mathbb{R}$, $\mathbf{v}_2$ defines a partition $(V_{\leq s}, V_{>s})$, where $V_{\leq s} = \{i | \mathbf{v}_2(i) \leq s\}$ and $V_{>s} = \{i | \mathbf{v}_2(i) > s\}$), and where for $i \in [|V|]$, we use $\mathbf{v}_2(i)$ to denote the i^{th} component of $\mathbf{v}_2$. Then, options `sign`, `bisection`, and `gap`, respectively, select s to be 0, the median of entries in $\mathbf{v}_2$, and the value that maximizes the gap of $(\text{argmin}(V_{>s}) - \text{argmax}(V_{\leq s}))$. The `ratio` option selects s to minimize the cut-ratio of the partition.

In the worst case, as shown in References [5, 77], the spectral partitioning method, regardless of the options, can fall behind the optimal solution by a sizable margin. In fact, Guattery and Miller showed that spectral partitioning with the first three options could fail badly even on some well-shaped 2D meshes [77].[15] In this regard, spectral partitioning methods are indeed heuristics. They have good real performance in spite of their worst-case limitations.

In the fall of 1993, Dan Spielman and I began to explore spectral methods. We wrote a MATLAB spectral embedding program using the eigenvectors $\mathbf{v}_2, \mathbf{v}_3$ associated with the second and third smallest eigenvalues of the Laplacian matrix to draw graphs (See Figure 2 for two illustrations of spectral embeddings). In other words, for a graph $G = (V, E)$ with eigenvectors $\mathbf{v}_2, \mathbf{v}_3$, for each node $u \in V$, we used $(\mathbf{v}_2(u), \mathbf{v}_3(u))$ as its two-dimensional coordinates. Then, for each edge $(u, u') \in E$, we embed this edge as a line segment from $(\mathbf{v}_2(u), \mathbf{v}_3(u))$ to $(\mathbf{v}_2(u'), \mathbf{v}_3(u'))$.

"The spectral embeddings of two-dimensional meshes are so planar!" we exclaimed.

While admiring the elegant spectral drawings, our minds were set on a conjecture we discovered through experimentation. It is well known that the Fiedler value of the $\sqrt{n} \times \sqrt{n}$ grid is $\Theta(1/n)$. In fact, Chung proved that the Fiedler value of any bounded degree graph with diameter D must be $O(1/D^2)$ [51]. It is also known that the Fiedler value of a complete binary tree with n nodes is $\Theta(1/n)$, a fact indicating that Chung's result is far from tight for the case of binary trees: the diameter of the complete binary tree is $\log n - 1$, and its Fiedler value is much smaller than $1/\log^2 n$.

Nevertheless, while they are planar graphs, trees are far from typical planar graphs. To get typical-looking low-diameter planar graphs, we added horizontal paths to connect nodes at each level of the tree (let us call the resulting graph an H-graph). The experimental measures were

[15]Consider a computer graphics model of a rectangular kite with two long tails. Spectral bisection will cut through the rectangular shape in a longer way.

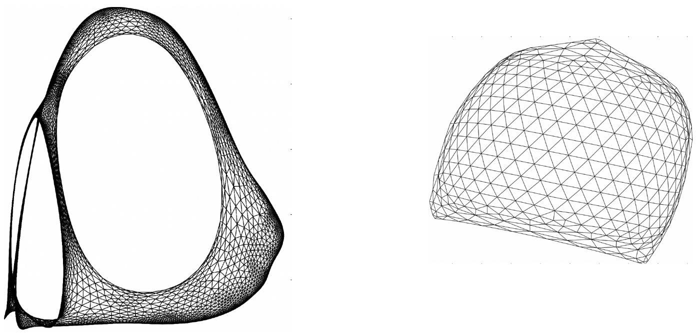

Fig. 2. Spectral embedding of planar graphs.

unambiguous: The Fiedler value of a H-graph of n nodes is $\Theta(1/n)$. So, we conjectured:

The Fiedler value for every bounded-degree planar graph of n *nodes is* $O(1/n)$.

This conjecture made us appreciate the work of Cheeger on the spectra of manifolds and their graph-theoretical extension by Alon and Milman [4], Jurrum and Sinclair [85], and Mihail [25]:

Theorem 3.1 (Cheeger Bound). *For any* $G = (V, E)$ *with* n *nodes and maximum degree* Δ,

$$\phi(G) \leq \sqrt{2 \cdot \Delta \cdot \lambda_2(G)}, \tag{4}$$

where $\lambda_2(G)$ *denotes the Fiedler value of* G *and* $\phi(G) = \min_{S \subset V} \text{cut-ratio}(S, V \setminus S)$ *denotes the isoperimetric number of* G. *Furthermore, the spectral partitioning method with the* `cut-ratio` *option finds a partition of* G *with cut ratio at most* $\sqrt{2 \cdot \Delta \cdot \lambda_2(G)}$.

Thus, if true, our "Planar-Fiedler-Value Conjecture" would take us a step closer to capturing the good performance of the spectral partitioning codes (with `cut-ratio` option) on well-shaped two-dimensional meshes. Notably, the well-shaped condition guarantees that all angles in the meshes are bounded below by a constant degree (e.g., 15 degrees). So, these meshes are bounded-degree planar graphs. Thus, the conjecture together with Cheeger's bound would establish that the spectral partitioning code is guaranteed to return a partition with cut-ratio $O(\sqrt{1/n})$. This bound matches the cut-ratio bound achieved by the classical Lipton-Tarjan planar separator theorem [103] (see below), which is the best possible for the class of bounded-degree planar graphs.

Theorem 3.2 (Lipton-Tarjan). *Every planar graph* G *with* n *nodes has a set* $S \subset V$ *of size* $O(\sqrt{n})$ *such that the removal of* S *divides the rest of the nodes into two disconnected sets* A *and* B *satisfying* $\max(|A|, |B|) \leq 2n/3$.

Thus,

$$\text{cut-ratio}(A \cup S, B) = \frac{\text{cut}(A \cup S, B)}{\min(|A \cup S|, |B|)} \leq \frac{\Delta \cdot |S|}{\min(|A \cup S|, |B|)} = O\left(\sqrt{1/n}\right).$$

Our conjecture, together with Cheeger's bound, could also help to explain the favorable performance of the spectral partitioning codes in VLSI design. In particular, graphs underlying Boolean circuits have small degrees. Because the number of layers in circuit layouts is usually much smaller than the number of logic gates in circuits and each layer has a planar layout, VLSI circuits are "nearly planar." So their Fiedler value should also be $O(1/n)$.

Cheeger's inequality also helped us to formulate a more ambitious high-dimensional conjecture for matching the performance of the spectral codes with that of the geometric partitioning codes on three and higher dimensional well-shaped meshes.

For any d, the Fiedler value of every n-node well-shaped mesh and nearest-neighbor graph in d dimensions is $O(1/n^{2/d})$.

We proved both these spectral conjectures three years later, and titled our paper, "Spectral partitioning works: Planar graphs and finite element meshes" [122]. The proof established a connection between the "energy" in geometric embeddings (in any dimension) of a graph $G = (V, E)$ and its Fiedler value:

$$\lambda_2(G) = \inf_{\sum_i \mathbf{u}_i = \vec{0}} \frac{\sum_{(i,j)\in E} ||\mathbf{u}_i - \mathbf{u}_j||^2}{\sum_{i=1}^{|V|} ||\mathbf{u}_i||^2}. \tag{5}$$

The analysis highlighted that both spectral partitioning and geometric partitioning work because of the same geometric properties intrinsic to planar graphs, nearest-neighbor graphs, and well-shaped meshes. The geometric methods explicitly use geometric data to exploit these properties [107, 108]. In contrast, the spectral methods recreate a geometric embedding from network data to extract these properties, and thus are powerful heuristics indeed.

The Cheeger-type characterization is an example of a powerful framework for studying algorithms and heuristics beyond worst-case analysis. We refer to it as the *condition-based framework* [135]. In this framework, we characterize input data x by two parameters: $(\text{size}(x), \text{condition}(x))$ where $\text{condition}(x)$ measures the *condition* of the input instance. For example, in Cheeger's bound, the $\text{condition}(G)$ is the Fiedler value $\lambda_2(G)$.

The condition-based framework has been more commonly used in numerical analysis and optimization. For example, the complexity of the Conjugate Gradient algorithm [72] to solve a symmetric positive-definite linear system, $Ax = b$, is bounded by

$$O(\text{size}(A)\sqrt{\kappa(A)}\log(1/\epsilon_{machine})),$$

where $\kappa(A)$ is the condition number of matrix A. Thus, for well-conditioned systems, the Conjugate Gradient algorithm can be much more efficient than its worst-case complexity. Similarly, the complexity of various interior-point algorithms for linear programming and gradient descent methods in machine learning can be better measured by the *condition-based framework* than the worst-case framework.

3.2 Modeling the Behaviors of Heuristics: Smoothed Analysis

During the 1980s and 1990s, many theoretical computer scientists were thinking about heuristics in computing, as highlighted in "Challenges for Theory of Computing: Report for an NSF-Sponsored Workshop on Research in Theoretical Computer Science (1999)" by Condon, Edelsbrunner, Emerson, Fortnow, Haber, Karp, Leivant, Lipton, Lynch, Parberry, Papadimitriou, Rabin, Rosenberg, Royer, Savage, Selman, Smith, Tardos, and Vitter:

"While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoffs, we are not done. In many situations, simple algorithms do well. Take for example the Simplex algorithm for linear programming, or the success of simulated annealing on certain supposedly intractable problems. We don't understand why! It is apparent that worst-case analysis does not provide useful insights on the performance of many algorithms on real data. Our methods for measuring the performance of algorithms and heuristics and our models of computation need to be further developed and refined. Theoreticians are investing increasingly in careful experimental work leading to identification of important new questions in the algorithms area. Developing means for predicting the performance of algorithms and heuristics on real data and on real computers is a grand challenge in algorithms."

The *Simplex Algorithm* is a family of iterative local search algorithms for solving optimization problems with a linear objective and linear constraints:

$$\textbf{maximize} \quad \mathbf{c}^T\mathbf{x} \quad \textbf{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}. \tag{6}$$

When given input data $(\mathbf{A}, \mathbf{b}, \mathbf{c})$, say with dimensions $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{c} \in \mathbb{R}^n$, it returns an n-dimensional vector $\mathbf{x} \in P_{\mathbf{A},\mathbf{b},\mathbf{c}} := \{\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ that optimizes $\mathbf{c}^T\mathbf{x}$, unless it determines that $P_{\mathbf{A},\mathbf{b},\mathbf{c}}$ is empty or unbounded along $\mathbf{c}$. In the feasible and bounded case, a simplex algorithm usually first finds a vertex of the convex polyhedron $P_{\mathbf{A},\mathbf{b},\mathbf{c}}$, and then iteratively moves to an adjacent vertex with a better objective value, until no such progress can be made.[16]

The simplex method for linear programming is truly an amazing heuristic in computing. Introduced in the 1940s by George Dantzig [55], the simplex method has since been refined and extensively implemented. Despite its exponential-time worst-case complexity [7, 95], it finds widespread use in industry. On real-world data, it demonstrates remarkable practical effectiveness. This raised several natural—and crucial—questions in the analysis of algorithms.

- "What is the difference between real-world data and worst-case data?"
- "What is the difference between real-world data and random data, as usually considered in average-case analyses?"
- "How should we model real behaviors of heuristics?"

During our analysis of the spectral partitioning method, Spielman and I were drawn to these questions. The spectral analysis of planar graphs also led us to high-dimensional convex polytopes and polyhedral graphs. After all, the celebrated Steinitz characterization states that the family of three-dimensional convex polyhedral graphs is the same as the family of three-vertex-connected planar graphs [128]. Our work at an Internet startup, Akamai Technologies, also got us close to real data, in particular, massive data from Content Delivery Networks and Internet maps.

Real-world data is neither worst-case nor random. Furthermore, real-world data usually has imprecision: physical measurements have imprecision; realizations of "perfect designs" have imprecision; most decisions have to deal with unexpected factors, which often introduce imprecision to the ideal plans. For example, in investment planning—one area where linear programs often arise—the stock value data is imprecise. Each time one looks them up, a company's stock value changes. The amount of change, as defined by market measurements, may appear random from minute to minute, but the expected stock values of that day are not totally random. They are correlated with the interplay between the financial health of companies and the overall investment environment.[17]

Just as a physical measurement such as the amount of protein in a slice of bread can be mathematically expressed as $p + \epsilon$, where ϵ is drawn from some distribution of measurement errors (e.g., Gaussian), the stock value could also be written as $s + \epsilon$, where ϵ is drawn from some distribution of market imprecision. In each case, neither the true amount of protein p nor the "true market value" s of the stock, is that random. This mathematical formulation of data provided us with a framework to model the behaviors of heuristics on real data in the presence of imprecision. For linear programming, our inputs $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ are now formulated as

$$(\mathbf{A}, \mathbf{b}, \mathbf{c}) = (\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}) + (\epsilon_{\mathbf{A}}, \epsilon_{\mathbf{b}}, \epsilon_{\mathbf{c}}), \tag{7}$$

where $\bar{\mathbf{A}}, \epsilon_{\mathbf{A}} \in \mathbb{R}^{m\times n}$, $\bar{\mathbf{b}}, \epsilon_{\mathbf{b}} \in \mathbb{R}^m$, and $\bar{\mathbf{c}}, \epsilon_{\mathbf{c}} \in \mathbb{R}^n$. $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ denote the ideal data, and $(\epsilon_{\mathbf{A}}, \epsilon_{\mathbf{b}}, \epsilon_{\mathbf{c}})$ denote errors in input data. We analyze the situation in which the simplex algorithm receives the

[16]Convexity then guarantees that every local optimizer is a global optimizer in linear programming.

[17]The concept that stock values are an interplay between a company's business health and market environments is more than just a theoretical idea. In practice, the magnitude of a stock's imprecision is often modeled by a stock's beta coefficient.

imprecise data $(\mathbf{A}, \mathbf{b}, \mathbf{c})$, rather than receiving the ideal data $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$. In our algorithm analysis framework, called *smoothed analysis*, we consider arbitrary ideal data (including the worst-case possibility), but the errors in the input data are assumed to be drawn from an error distribution D, such as a Gaussian [121]. To capture the impact of perturbations, without loss of generality, we assume that data $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ are scaled to have unit norms. For example, suppose that $T(\mathbf{A}, \mathbf{b}, \mathbf{c})$ represents the time complexity of a simplex algorithm to solve the linear program given by the data $(\mathbf{A}, \mathbf{b}, \mathbf{c})$.

Then, in the smoothed analysis framework, we measure the *smoothed time complexity* of this simplex algorithm by

$$\sup_{(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})} \mathbb{E}_{(\epsilon_{\mathbf{A}}, \epsilon_{\mathbf{b}}, \epsilon_{\mathbf{c}})} \left[T(\mathbf{A}, \mathbf{b}, \mathbf{c}) \right]. \tag{8}$$

We proved the following theorem [121]:

Theorem 3.3 (Smoothed Analysis of the Simplex Method). *The smoothed time complexity of the simplex algorithm (with shadow-vertex pivoting rule) is polynomial in m, n, and $\frac{1}{\sigma}$ where n and m, respectively, denote the dimensions and the number constraints of the linear program, and σ denotes the variance of the Gaussian noises in data.*

Assuming that imprecision and noise play a role in real-world linear programs, this theorem establishes that the simplex method, despite its exponential worst-case complexity, usually runs in polynomial time. Thus, its polynomial-time smoothed complexity may partly explain the practical effectiveness of this classical method. Furthermore, it demonstrates that the exponential-time worst-case instances, as classically constructed by Klee and Minty [7, 95], are intrinsically brittle.

Smoothed analysis has been extended to other commonly used heuristics that exhibit poor worst-case performance, such as the k-means method in data mining [15] and local search for optimization [11]. While these heuristics may not be perfect, they are often appreciated for their intuitive and elegant nature and tend to perform well in the smoothed model. In fact, many of the worst-case instances constructed in theoretical computer science are inherently fragile. For example, in the worst-case scenario, the k-means method (also known as Lloyd's method) can require an exponential number of iterations to converge to the final clustering. These worst-case instances are meticulously designed, necessitating exponential precision in positioning data points in space, to restrict the progress of the k-means method to an exponentially small improvement in each iteration. Arthur, Manthey, and Röglin demonstrated that worst-case constructions under the noise model rarely produce data instances where the k-means method exhibits superpolynomial behavior [15].

Subsequently, Angel, Bubeck, Peres, and Wei [11] established that Local Max-Cut, which is a complete problem for the complexity class **PLS (Polynomial-time Local Search)** [86], has a polynomial smoothed complexity. Their work was motivated by the statement of Johnson, Papadimitriou, and Yannakakis, who originally defined the computational class PLS:

> "*Practically all empirical evidence suggests that finding locally optimal solutions is much easier than solving NP-hard problems.*"

Machine learning has also proven to be fertile ground for smoothed analysis [1, 29, 30, 41, 54, 70, 79, 80, 88, 89, 112]. Since data in computational learning theory is typically drawn **independently and identically distributed (i.i.d.)** from some possibly unknown distribution, traditional average-case analysis has already been applied to learning algorithms. However, simple learning tasks—such as PAC-learning DNFs and agnostically learning decision trees—can be computationally challenging even for uniform distributions. For learning problems where concept classes, (deep) learning architectures, and distribution structures are themselves susceptible to noise and imprecision, smoothed analysis complements traditional analysis of learning algorithms.

For example, Ge, Huang, and Kakade [70] proved that under the smoothed model, where the parameters are randomly perturbed from an adversarial starting point, there exists a polynomial-time algorithm (using a polynomial number of samples) for learning mixtures of Gaussians in high dimensions. The algorithm employs a novel approach to decompose the moment tensor of the Gaussian mixture. Tensor decomposition is instrumental for analyzing high-dimensional data, both in data mining and machine learning, and serves as a foundational technique in smoothed analysis of various learning tasks [29, 30].

In another line of examples, the classical sparse Fourier algorithm for PAC-learning DNFs and agnostically learning decision trees has been shown to be polynomial-time implementable when the underlying product distributions have a small degree of noise [88]. The proof is built on a structural property of the Fourier spectrum of any Boolean function over perturbed product distributions. It was further shown by Abbe, Boix-Adsera, Brennan, Bresler, and Nagaraj [1] that this smoothed analysis of the Fourier spectrum establishes that a decision tree with depth $\log(n)$ over n variables probably satisfies what they referred to as the general staircase property. This, in turn, implies that regular neural networks can learn decision trees in the smoothed product distribution model [1].

In smoothed online learning settings, Rakhlin, Sridharan, and Tewari [112] demonstrated that even a small amount of noise can render function classes with infinite Littlestone dimension learnable. Haghtalab, Roughgarden, and Shetty [79, 80] further extended smoothed analysis to encompass more general online and differentially private learning models, including those with adaptive adversaries. They proved that instead of the Littlestone dimension, the regret and privacy error bounds depend solely on the VC dimension, the magnitudes of the perturbations (to adversaries' choices) and a geometric covering measure of the underlying hypothesis class.

In complexity theory, some problems are known to be "difficult on average." Well-known examples of such problems are the lattice problems—with the shortest vector problem as a prominent example—whose average-case hardness has been the basis for their cryptographic applications. Likewise, some intractable problems remain intractable in certain smoothed settings.

Beier and Vöcking [24] elegantly characterized the conditions under which NP-complete binary optimization problems[18] exhibit smoothed polynomial-time complexity. The intractable direction of their result, along with the result of Reference [11] that the PLS-complete Local Max Cut is tractable in the smoothed model, provides theoretical evidence for the aforementioned Johnson-Papadimitriou-Yannakakis statement regarding the inherent complexity-theoretical difference between local search and global optimization. In algorithmic game theory, Chen, Deng, and Teng [50] proved that computing Nash equilibria and Arrow-Debreu market equilibria remains PPAD-hard, highlighting another sharp contrast between PPAD-complete problems and PLS-complete problems. Concerning learning in neural networks, Daniely, Srebro, and Vardi [54] demonstrated that learning depth-3 ReLU networks under the Gaussian distribution remains intractable in the smoothed framework when random noise is added to the network's parameters (under a cryptographic assumption).

These hardness results refine complexity characterizations that underscore the fundamental computational challenges across diverse problem domains, from optimization to game theory to

[18]Each binary optimization problem over n Boolean variables is defined by a linear objective function over $\{0, 1\}^n$ and a set of linear constraints.

machine learning. They also provide strong motivation for the development of a new generation of *beyond-worse-case theoretical frameworks* to identify sharper criteria for distinguishing between intractable and tractable instances, given the practical importance of these computational problems.

4 DESIGNING HEURISTICS AND FORMULATING "HEURISTIC THEORIES"

Designing heuristics can be fun, even for a theoretician. And finding mathematical ways to capture certain facets of these heuristics can be equally exciting.

4.1 High-dimensional Medians: An Evolutionary Algorithm

Implementing a geometric mesh partitioning package at Xerox PARC [71] provided me with my first opportunity to design and use a heuristic, to deal with an unexpected gap between theory and practice. It started as an "intuitive and effective, yet without theoretical guarantees" kind of heuristic, as described by David Applegate. But ultimately, it inspired the development of a theory to "heuristically" explain its practical effectiveness, as well as a new polynomial-time approximation algorithm for a basic concept in robust statistics.

Once John Gilbert and I began to code up the geometric mesh partitioning algorithm [107, 108] in MATLAB, we realized that all but one step could be implemented by standard techniques. Ironically, the difficult step has a constant time complexity according to computational complexity theory, i.e., time complexity independent of the mesh size.

This step concerns the computation of a *generalized median*—known as a *centerpoint* [62]—in four dimensions. Recall that for a set of numbers $P = \{p_1, \ldots, p_n\}$ and positive $\beta \leq 1/2$, we call a number $c \in \mathbb{R}$ a *β-median* of P if $\max(|\{i : p_i < c\}|, |\{j : p_j > c\}|) \leq (1 - \beta) \cdot n$. The centerpoints are "medians" in high dimensions.

Definition 4.1 (Centerpoints). Suppose $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is a point set in $\mathbb{R}^d$. For a positive $\beta \leq 1/2$, a point $\mathbf{c} \in \mathbb{R}^d$ is a β-center P if for all unit vectors $\mathbf{z} \in \mathbb{R}^d$, the projection $\mathbf{z}^T\mathbf{c}$ is a β-median of the projections, $\{\mathbf{z}^T\mathbf{p}_1, \ldots, \mathbf{z}^T\mathbf{p}_n\}$.

For fixed dimensions, centerpoints can be approximated efficiently using the sampling theory developed by Vapnik and Chervonenkis [138, 139], which I learned from CMU's computational learning theory class. The reason is the following: The region of centerpoints is a convex polytope defined by the intersection of the set of halfspaces, each of which has a complement containing fewer than $\frac{n}{d+1}$ points from P. Because the family of halfspaces in d-dimensions has VC dimension $d + 1$, one can use the centerpoint of a small sample as an effective approximation for the whole.

All of this sounded great in theory until I planned to write this part of the code.

"We might need a CPLEX license,"[19] I told John.

But the next day, I was more dejected after a back-of-the-envelope calculation: By the theory of Vapnik and Chervonenkis [139], we would need a sample of $s = \frac{d}{\epsilon^2} \log \frac{d}{\epsilon}$ points. In our case, $d = 4$ and four points in $\mathbb{R}^4$ define a hyperplane. To formulate the linear program to find a centerpoint of the sample, we would first need to construct $\binom{s}{4}$ hyperplanes. Every point set in $\mathbb{R}^4$ has a 1/5-centerpoint. So ϵ should be less than 1/5. Suppose that we set $\epsilon = 1/10$, then $s \approx 2{,}500$. The number of hyperplanes would be $\binom{2500}{4}$, a startling 10^{12}.

[19]CPLEX is a linear programming and optimization package developed by Robert Bixby developed in 1988 and is a product of CPLEX Optimization Inc. It uses the simplex method to solve linear programs.

"Well, the VC bound is just a theory. It's possible that a smaller sample, say $s = 100$, might still perform well in practice," I hoped.

"But wait, even so, we still need to examine about four million hyperplanes."

It suddenly dawned on me why heuristics are used in the practice of computing and why many theoretically sound algorithms are not!

"To implement our 'provably good' mesh partitioning algorithm, I am going to use a heuristic step," I said sheepishly to John.[20]

Many heuristics, including spectral partitioning methods, were built on intuitions originating from classical theoretical work. Likewise, the intuition behind our four-dimensional centerpoint heuristic was derived from the following classical theorem by Radon in convex geometry [56, 137].

Theorem 4.2 (Radon). *Every set $P \subset \mathbb{R}^d$ of at least $d + 2$ points can be divided into two subsets (P_1, P_2) such that the convex hull of P_1 and the convex hull of P_2 have nonempty intersection.*

The points in this intersection are called the *Radon points* of P. The base case ($d = 1$) of this theorem simply states that the median of any three numbers is in the interval of the other two, and is their Radon point. In high dimensions, a Radon point of $d + 2$ points is a $\frac{1}{d+1}$-centerpoint of the set. Radon's proof yields an $O(d^3)$ time algorithm for computing a Radon point. Our four-dimensional centerpoint heuristic applies Radon's algorithm as the natural selection process in an evolutionary algorithm [52, 71]: Four random samples pool their data together to create an offspring, namely, their Radon point. This process, see Algorithm 1 below, is designed to enhance their "going-to-the-center" fitness.

ALGORITHM 1: RadonEvolution(P, d, h)

```
1: Select a random population $S$ of $(d + 2)^h$ samples from $P$.
2: while $S$ has more than one point do
3:     Randomly partition $S$ into groups of $(d + 2)$, and replace each group by its Radon point.
4: Return the final point $\mathbf{c}$ left in $S$.
```

At Xerox PARC, Gilbert taught me how to implement Radon's algorithm in MATLAB utilizing its built-in QR-decomposition function. For $d = 4$, it was super-fast. Experimentally, our little "genetic algorithm" was a remarkable success, far surpassing my expectations and understanding. We designed a heuristic "intuitive and effective yet without theoretical guarantees" to support our otherwise "provably good" mesh partitioning algorithms. The excellent experimental data encouraged us to continue to think about the worst-case quality of this RadonEvolution process. It only took two more years and a few more collaborators before we finally proved the following worst-case mathematical guarantee: Let $\beta_d = 1/d^2$. For any point set $P \in \mathbb{R}^d$, with probability $1 - \delta$, RadonEvolution computes a β_d-centerpoint of P, using $(d \log \frac{1}{\delta})^{2 \log d + 3}$ time.

One may note that the quality guarantee of $1/d^2$-centerpoints has a quadratic gap from the existence of $\frac{1}{d+1}$-centerpoints. However, like Cheeger's quadratic spectral gap for isoperimetry

[20]Years later when I told him about this part of my career, my former Ph.D. student Kyle Burke responded, "In theoretical computer science, papers don't use heuristics, but code does." Kyle's statement made me appreciate Karmarkar's "secret sauce" in the implementation of his landmark polynomial-time linear programming algorithm at AT&T (see Reference [115]).

and conductance, this result provides a nontrivial polynomial guarantee. In fact, further improvement to RadonEvolution also led us to the first polynomial-time approximation algorithm for centerpoints in arbitrary dimensions [52]. These mathematical analyses have not only helped us to partially explain the experimental success of the RadonEvolution heuristic, but have also helped me to appreciate the power of natural selection, and the metaheuristics that it inspires, such as evolutionary algorithms and genetic algorithms.

4.2 Multiscale Sampling in Big Data

In the late 1990s, PageRank emerged as a powerful tool for Web search [39, 99]. This linear algebraic concept, when applied to the Web graph, provided numerical ranking of Web sites as measures of their importance/relevance. Mathematically, for a (weighted) directed graph $G = (V, E)$ of n nodes and for a parameter $0 < \alpha \leq 1$, the PageRank $\mathbf{p}^{G,\alpha} \in \mathbb{R}^n$ satisfies the following linear equation:

$$\mathbf{p}^{G,\alpha} = \alpha \cdot \frac{\mathbf{1}}{n} + (1-\alpha) \cdot \mathbf{p}^{G,\alpha}\mathbf{M}_G, \tag{9}$$

where $\mathbf{M}_G = \mathbf{D}_G^{-1}\mathbf{A}_G$ is the *random-walk matrix*, and $\mathbf{1}$ denotes the n-dimensional row vector of all 1s. So $\frac{\mathbf{1}}{n}$ represents the uniform distribution over V. It is well known that $\mathbf{p}^{G,\alpha}$ (also a row vector) specifies a distribution over V, modeling the stationary distribution in a Markov process that combines random walks (with probability $1-\alpha$) and random jumps (with probability α) [39, 110].

In the early days, Google crawled the Internet at night to update the Web graph and its PageRank to prepare for new Web searches. As the Web graph grew in size, it began to update the PageRank values for significant sites more regularly than others. This raises several natural questions.

> *Can one efficiently identify significant nodes without exploring the entire network? Can one efficiently update the PageRank of significant nodes without visiting the entire network?*

Heuristics have become increasingly necessary in the age of Big Data. For some applications, even linear-time computation may be too slow. In addition, many computational tasks require making decisions before accessing the entire dataset. In traditional large-scale statistical analysis, as in the social and medical sciences, sampling also plays a crucial role. Sampling is fundamental to the foundation of statistical and computational learning theory as well. Over the years, a substantial amount of theory has been developed to capture the capabilities and limitations of these statistical methodologies. As data and decision-making processes become more complex, the need for more sophisticated sampling and exploration techniques naturally arises.

We now return to the two questions concerning large-scale PageRank approximation. Let us try to formulate a computational problem to capture these two questions. For a parameter $\Delta \in \mathbb{R}$, call a node Δ-*significant* in G if its PageRank value is at least Δ. Note that by Equation (9), the minimum PageRank value is at least $\frac{\alpha}{n}$. Furthermore, PageRank is a distribution over n nodes and the total PageRank value is 1. Thus, the average PageRank value is $\frac{1}{n}$ and the number of Δ-significant nodes is at most $\frac{1}{\Delta}$. So, let us be brave by asking an optimistic question about computing Δ-significant nodes without exploring the entire network.[21]

> *Can one identify a set containing all Δ-significant nodes and approximately compute their PageRank values in $\tilde{O}(\frac{1}{\Delta})$ time, for all $\frac{\alpha}{n} < \Delta < 1$?*

If we could magically sample a node in a network with probability equal to its PageRank value, then the expected number of draws to see a particular Δ-significant node would be $O(\frac{1}{\Delta})$. Then, by

[21]Below, in addition to constants and lower order terms, $\tilde{O}$ also hides poly-logarithmic factors of the input size.

the union bound, a sample of $O(\frac{1}{\Delta}\cdot\log\frac{1}{\lambda\Delta})$ nodes, i.i.d. drawn according to the PageRank distribution, would contain all Δ-significant nodes with probability $1-\lambda$. But this could be a chicken-and-egg problem. *How should one sample nodes according to their PageRank values without knowing the PageRank?*

The personalized PageRank introduced by Haveliwala [82] provided an initial step for a heuristic approach. For any $v \in V$, the personalized PageRank associated with v is the solution of the following equation, where $\mathbf{e}_v$ is the unit row vector whose vth entry is 1:

$$\mathbf{p}_v^{G,\alpha} = \alpha\cdot\mathbf{e}_v + (1-\alpha)\cdot\mathbf{p}_v^{G,\alpha}\mathbf{M}_G. \tag{10}$$

Crucially, personalized PageRank is an approximately "local" concept, highlighted by the following Taylor series. Note below that $\mathbf{e}_v\cdot\mathbf{M}^k$ denotes a random walk of k steps that starts at v:

$$\mathbf{p}_v^{G,\alpha} = \alpha\sum_{k=0}^{\infty}(1-\alpha)^k\cdot\mathbf{e}_v\cdot\mathbf{M}_G^k. \tag{11}$$

This Taylor series captures the following random-walk-based sampling process for personalized PageRank: imagine a traveler has a base station at v and takes random walks to explore the network. The traveler has a camera that can take only one picture and a random coin that lands as "tails" with a probability of α. At each node, the traveler flips the coin. If it lands heads, then the traveler moves forward with random walks; otherwise, the traveler takes a picture of the current node, returns to the base station, and offloads the picture to the base station. In the limit, the empirical distribution of the images collected at the base station is exactly the personalized PageRank $\mathbf{p}_v^{G,\alpha}$. This observation led to a random-walk-based sampling process to approximate personalized PageRank to a desired precision, with the following guarantee [34]:

Lemma 4.3 (Personalized PageRank Approximation). *For any $0<\epsilon,\delta,\rho<1$, one can compute in time $\tilde{O}(\frac{1}{\epsilon}\cdot\frac{1}{\rho^2})$ a vector $\tilde{\mathbf{p}}_v^{G,\alpha}$ such that with probability at least $1-\delta$:*

$$(1-\rho)\cdot\mathbf{p}_v^{G,\alpha} - \epsilon\cdot\mathbf{1} \le \tilde{\mathbf{p}}_v^{G,\alpha} \le (1+\rho)\cdot\mathbf{p}_v^{G,\alpha} + \epsilon\cdot\mathbf{1}. \tag{12}$$

Note that empirical personalized PageRank under the uniform distribution is an unbiased estimator of PageRank, because by linearity we have

$$\mathbf{p}^{G,\alpha} = \frac{1}{n}\left(\sum_{v\in V}\mathbf{p}_v^{G,\alpha}\right). \tag{13}$$

However, the PageRank approximation using Lemma 4.3 highlights the following "practical reality" in data science:

Data with higher precision requires a higher cost to obtain.

This is particularly true in statistical sampling—as in Lemma 4.3—because more precise estimation requires larger samples. In Lemma 4.3 for the personalized PageRank approximation, to be ϵ precise, one needs to take $\Omega(\frac{1}{\epsilon})$ random walks. So, in $\tilde{O}(\frac{1}{\Delta})$ time, one can only approximate $\tilde{O}(1)$ personalized PageRank to $\Theta(\Delta)$ precision. But obtaining an adequate estimation of the Δ significant PageRanks requires approximately computing personalized PageRank vectors for a significantly larger number of random nodes than just $\tilde{O}(1)$.

Multilevel thinking led us to the following heuristic for balancing precision and cost. For any positive $\beta\le 1$, let us consider $\log_{1+\beta}\frac{1}{\beta\Delta}$-precision levels: $\epsilon_h = \frac{1}{(1+\beta)^h}$, for $h\in[1,2,\ldots,\log_{1+\beta}\frac{1}{\beta\Delta}]$. At each level, the maximum number of personalized PageRanks is sampled within the budgeted $\tilde{O}(\frac{1}{\Delta})$ time. In other words, at level ϵ_h, $\tilde{O}(\frac{1}{\Delta})\cdot\epsilon_h$ personalized PageRanks are sampled. A natural estimator is then built from these personalized PageRanks.

As it turned out, this heuristic provides a bicriteria mathematical guarantee [34]:

THEOREM 4.4 (MULTISCALE SIGNIFICANT PAGERANK APPROXIMATION). *For any graph $G = (V, E)$, parameters $0 < \alpha < 1$, $\frac{\alpha}{n} < \Delta < 1$, $0 \leq \beta \leq 1$, in $\tilde{O}(\frac{1}{\alpha\beta\Delta})$ time, the above heuristic outputs a set $S \subset V$ such that S contains all Δ significant nodes but does not contain any nodes whose PageRank value is less than $(1-\beta)\cdot\Delta$. Moreover, it estimates the PageRank value of every node in S to within $[(1-\beta/2)\cdot\Delta, (1+\beta/2)\cdot\Delta]$.*

Note that the full set of personalized PageRank vectors for an *n*-node graph G = (V, E), as we denoted before as $\{\mathbf{p}_1^{G,\alpha}, \ldots, \mathbf{p}_n^{G,\alpha}\}$, forms an $n \times n$ matrix:

$$\mathbf{PPM}^{G,\alpha} = \begin{bmatrix} \mathbf{p}_1^{G,\alpha} \\ \vdots \\ \mathbf{p}_n^{G,\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^{G,\alpha}(1) & \ldots & \mathbf{p}_1^{G,\alpha}(n) \\ \vdots & \vdots & \vdots \\ \mathbf{p}_n^{G,\alpha}(1) & \ldots & \mathbf{p}_n^{G,\alpha}(n) \end{bmatrix}. \tag{14}$$

For a graph $G = (V, E)$, $\mathbf{PPM}^{G,\alpha}$ is referred to as its *personalized PageRank matrix*. This matrix, which is completely dense whenever G is strongly connected, has several interesting structural properties [136]. Let us mention a few basic ones here. First, its row sum is the column vector with all 1s, and hence $\mathbf{PPM}^{G,\alpha}$ defines a Markov chain. In fact, this PageRank-based Markov chain and G's random-walk Markov chain have the same stationary distribution. Second, its column sum is precisely the PageRank vector of G. Third, $\mathbf{PPM}^{G,\alpha}$ defines a detailed balance Markov chain if and only if G is an undirected graph. The personalized PageRank matrix, as depicted in Equation (14), also provides a helpful visualization of significant PageRank approximation heuristics. For each $v \in [n]$, its PageRank is the sum of its column in the matrix:

$$\mathbf{p}_1^{G,\alpha}(v) + \mathbf{p}_2^{G,\alpha}(v) + \cdots + \mathbf{p}_n^{G,\alpha}(v).$$

In the sum, for each $u \in V$, $\mathbf{p}_u^{G,\alpha}(v)$ is u's *contribution* to the PageRank of v. Conversely, one may also view v's PageRank as v's *influence power* in the network G, and $\mathbf{p}_u^{G,\alpha}(v)$ as v's *influence* on u. In this regard, the columns of the personalized PageRank matrix capture each given node's forward influence to others, while the rows capture the reversed direction measuring how much others influence each given node. Since the latter can be approximated through a local process (based on truncated random walks), the significant PageRank approximation algorithm primarily emphasizes the reversed row perspective of "being influenced" in its overall process to estimate each node's influence, as opposed to the column perspective of forward influence.

This "reversed diffusion process," introduced in our significant PageRank algorithm, has proven to be a fundamental technique for enhancing the efficiency of previous solutions for the influence maximization problem [33]. The earlier approach of Kempe-Kleinberg-Tardos [93] focused on seed sets and their natural "forward influence" *along* the direction of the cascade. The reversed diffusion process centers around a randomly selected target node and explores the "being-influenced-by" relationship *against* the direction of the cascade. This reversed diffusion process emulates the approximation of personalized PageRanks. Borgs, Brautbar, Chayes, and Lucier [33] applied this reversed diffusion process to design the first scalable algorithm for influence maximization in independent cascade and linear threshold models. Tang, Shi, and Xiao [130, 131] made a further improvement for the triggering model.

4.3 Local Clustering in Massive Networks

As networks become massive, techniques that prioritize data localities for network analysis become instrumental. This is particularly true when the network data—such as the Web, social networks, and real-time movement data—are no longer stored within the same data center. Local network analysis methods aim to identify significant patterns[22] or find solutions without examining the entire data set. Thus, they are wonderful subjects for heuristic designs. In the following, we consider a basic *local-network-exploration* framework for analyzing big data. We use this framework to discuss an example of heuristic design for network clustering. This example also helps illustrate the formulation of a "heuristic theorem" for capturing the performance of these clustering heuristics.

In this *local network analysis framework* [135], the underlying network $G = (V, E)$ is considered to be "hidden." The input to an exploration algorithm is either a starting node $s \in V$ or a starting subset $S \subset V$ of G, typically with $|S| \ll |V|$. The network analysis algorithm is called a *local algorithm* if at each step it only examines nodes it has seen before or nodes connected to those it has seen before. For example, the random-walk-based local PageRank approximation algorithm (discussed in the previous subsection) is a local algorithm in this framework, as are the classical **breadth-first search (BFS)** and **depth-first search (DFS)** algorithms.

The distinction between local and global network analysis is subtle. Local network exploration, when given sufficient time, could visit the entire network and thus becomes a global method. Thus, local and global network analyses only differ when the former requires decision-making before covering the entire network. For instance, in AI search, breadth-first search, depth-first search, and their extensions are often considered as local techniques for strategic decision making over large game trees. In contrast, for algorithm design, BFS and DFS are typically employed as global algorithmic techniques for graphs of moderate size. The former needs to make decisions that are "local to" the root without evaluating the entire game tree, whereas the latter allows algorithms to run in polynomial time relative to the graph's size. This difference might be the reason why, during my time as a Ph.D. student, AI search methods were taught as heuristics in AI classes, whereas BFS and DFS were presented as algorithmic techniques in theory classes. Nowadays, with the proliferation of massive data and networks, the distinction between local and global algorithms is becoming more pronounced.

Local algorithms must make some basic "heuristic decisions":

- How long should a local algorithm run?
- How much data can it explore?

As a result, it is more subtle to measure the performance of local algorithms than of global ones.

- How should one define the size of the problem, when local algorithms need to make decisions with only partial access to input data?

Let us return to our example of local clustering. The task is to identify a ***good*** cluster ***near*** a starting vertex $v \in V$ in a ***hidden*** network $G = (V, E)$. More concretely, we aim to achieve the following aspired goal for local clustering: Given a node $v \in V$ in a network $G = (V, E)$ and a target conductance ϕ, find a cluster S containing v such that $\text{conductance}_G(S) \leq \phi$.

For general G and ϕ, it is NP-hard to decide whether G has a cluster S (containing v) such that $\text{conductance}_G(S) \leq \phi$. Thus, even in the global framework, approximations or heuristics are required. Local clustering potentially faces more challenges. Here again, classical theoretical work provides valuable guidance for heuristic design. Cheeger's spectral inequality states that for any undirected graph $G = (V, E)$:

$$\frac{\phi^2}{2} \leq \lambda_2 \leq 2 \cdot \phi, \tag{15}$$

[22]Identifying meaningful patterns in massive networks is a fundamental family of graph theoretical problems in data mining and knowledge discovery.

where $\phi = \min_{S \subset V} \text{conductance}(S)$ and λ_2 denotes the second smallest eigenvalue of the *normalized Laplacian matrix* $\mathcal{L}_G = \mathbf{D}_G^{-1/2}(\mathbf{D}_G - \mathbf{A}_G)\mathbf{D}_G^{-1/2}$.

Let us first put this inequality in an algorithmic perspective through the lens of the global framework to set up the local heuristics. Consider a cluster $S \subset V$ in G. By the right-hand inequality of Equation (15), one can conclude that $\lambda_2 \leq 2 \cdot \phi \leq 2 \cdot \text{conductance}(S)$. Then, by the left-hand inequality of Equation (15) and its analysis, one concludes that the spectral partitioning method would find another cluster S' (in polynomial time) satisfying the following quality:

$$\text{conductance}(S') \leq \sqrt{2\lambda_2} \leq 2\sqrt{\text{conductance}(S)}.$$

In other words, in the global algorithmic framework, if nature promises us that G has a good cluster S, then without needing any information about S, we can use the spectral partitioning method to find another S', which may or may not be as good as S, but at least satisfies $\text{conductance}(S') \leq 2\sqrt{\text{conductance}(S)}$. We asked a related question:

> *In the local framework, if nature promises us that there is a good cluster S containing $v \in V$, then can we efficiently find a "good enough" cluster S' containing v? More precisely, do we have a local clustering method that can find a cluster S', in time linear or polynomial in $\tilde{O}(|S'|)$, satisfying* $\text{conductance}(S') \leq \tilde{O}(\sqrt{\text{conductance}(S)})$*?*

The first local clustering method `Nibble`, designed by Spielman and Teng [125], used the following basic algorithmic scheme from Cheeger's analysis: (1) compute a desirable vector $\mathbf{v} \in \mathbb{R}^{|V|}$ from the input graph $G = (V, E)$. (2) Order the nodes in V by sorting the entries in $\mathbf{v}$, say in descending order. Let $\pi_{\mathbf{v}}$ denote the ordering on V and let $S_k = \{\pi_{\mathbf{v}}(1), \ldots, \pi_{\mathbf{v}}(k)\}$. Return S_{k^*} where $k^* = \text{argmin}_k \text{conductance}(S_k)$. We call step (2) $\text{Sweep}(G, \mathbf{v})$.

In the analysis of Cheeger's inequality, the spectral partitioning method uses $\text{Sweep}(G, \mathbf{v}_2)$, where $\mathbf{v}_2$ is the Fiedler vector of the (normalized) Laplacian. `Nibble` was built on the mathematical intuition underlying Lovász-Simonovits' analysis of **Markov chain Monte Carlo (MCMC)** [104]. In what could be considered a local analogue to Cheeger's inequality, Lovász and Simonovits showed that if the random-walk starting at a node v does not converge rapidly to the stationary distribution of the random-walk Markov chain, then there must be some local bottleneck near v. In other words, some of these initial random-walk distributions can be used to identify a cluster with small conductance close to v.

However, while the random walks may converge slowly, the support of random-walk distribution can grow rapidly. Central to `Nibble` is the control of sparsity in random-walk distributions. In addition to the graph $G = (V, E)$ and the starting node $v \in V$, `Nibble` takes two additional inputs: ϕ, the target conductance, and T, a parameter for time complexity allowed by local exploration. $\texttt{Nibble}(G, v, \phi, T)$ computes $k = \tilde{O}(\phi^{-2})$ *sparse* vectors $\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_k$ in $\mathbb{R}^{|V|}$, and returns the cluster with smallest conductance among $\text{Sweep}(G, \tilde{\mathbf{u}}_1), \ldots, \text{Sweep}(G, \tilde{\mathbf{u}}_k)$. Mathematically, $\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_k$ are *sparse* approximations of the first k-step random-walk distributions starting at v: $\mathbf{u}_1 = \mathbf{e}_v \cdot M_G$, $\ldots$, $\mathbf{u}_k = \mathbf{e} \cdot M_G^k$. To limit the running time of the local clustering algorithm to $\tilde{O}(T \cdot \phi^{-4})$, the approximation precision at each reachable node $u \in V$ is set to $\epsilon_u = \frac{\phi^2}{T} \cdot d_u$, where d_u is the degree of u' in G.

Compared to typical graph algorithms in theoretical computer science, `Nibble` is undoubtedly a heuristic in the traditional sense. For one, it offers no theoretical worst-case guarantee: There exists $G = (V, E)$, $v \in V$, and $\phi < 1$, such that G has a cluster S containing v satisfying $\text{conductance}(S) \leq \phi^2$, but $\texttt{Nibble}(G, v, \phi, T)$ fails to compute a cluster S' containing v satisfying $\text{conductance}(S') = \tilde{O}(\phi)$ even with $T \gg |S|$. In general, the selection of ϕ and T requires additional local information gathering. However, the practical effectiveness and efficiency of `Nibble` were demonstrated by experiments on protein networks by Voevodski et al. [141].

As part of our work for scalable Laplacian solvers and spectral graph sparsification [120, 125], Spielman and I formulated a "heuristic theorem" that captures the *statistical effectiveness* of `Nibble`. Recall that $\mathrm{vol}_G(S)$ denotes the total degree of nodes in S.

Theorem 4.5 (Statistical Performance of Local Clustering). *For any cluster $S \subset V$ in $G = (V, E)$, there exists a subset $S_g \subseteq S$ satisfying $\frac{\mathrm{vol}_G(S_g)}{\mathrm{vol}_G(S)} \geq \frac{1}{2}$ and $\forall \phi \geq \sqrt{\mathrm{conductance}(S) \log^3 n}$, for all $v \in S_g$, there exists $b \in [1, 2, \ldots, \log(\mathrm{vol}(S))]$ such that* `Nibble`$(G, v, \phi, 2^b)$ *finds a cluster C with* $\mathrm{conductance}(C) \leq \phi$. `Nibble` *is* locally scalable *in that* `Nibble`(G, v, ϕ, T) *runs in time $\tilde{O}(\frac{T}{\phi^4})$.*

This theorem acknowledges the reality that, for some cluster $S \subset V$, a local clustering starting from certain nodes in S may not succeed. However, it shows that if one selects a random node from S with probability proportional to its degree, then with probability at least 50 percent, the local clustering algorithm `Nibble` will succeed in finding a cluster C with $\mathrm{conductance}(C) \leq \tilde{O}(\sqrt{\mathrm{conductance}(S)}$ (that is, a Cheeger-type of guarantee) in time nearly linear in $\mathrm{vol}(S)$.

Several local clustering heuristics have now been developed that have further improved this theorem. Andersen, Chung, and Lang [8] proved that a sparse approximation of the personalized PageRank vector can be used in Sweep to improve the quality of the local cluster to $\sqrt{\mathrm{conductance}(S) \log n}$, and reduce time complexity to $\tilde{O}(T \cdot \phi^{-1})$. Subsequently, Andersen and Peres [10] (further improved in Reference [9]) discovered a new local clustering process, based on a notion of evolving sets, to obtain an almost optimal local clustering algorithm, improving the quality of the local cluster to $O(\sqrt{\mathrm{conductance}(S)})$ and time dependency to nearly linear in $\tilde{O}(T \cdot \phi^{-0.5})$.

This "heuristic" theorem is indeed of significant use in theory. A variation of this theorem provided an instrumental tool for designing the first scalable algorithms for spectral graph sparsification and for solving Laplacian linear systems [124, 126]. In fact, this Laplacian solver uses low-stretch spanning trees [2, 3, 63] and spectral sparsification to support a multilevel method generated by partial Gaussian eliminations.

5 CHARACTERIZATION OF HEURISTIC SOLUTIONS: AXIOMATIC APPROACHES

In practice, data has not only grown in size but also in dimensionality. Therefore, techniques for effective dimensionality reduction have become increasingly essential.

5.1 Dimensionality of Big Models

In most applications, the ambient space of the data usually has dimension higher than that of the input data. For example, while many graphs in scientific computing and network sciences are sparse—e.g., an n-person Facebook graph with $\tilde{O}(n)$ friend relations—the ambient space for n-node graphs has $\Theta(n^2)$ dimensions. In some applications, the natural ambient space for data models or their solutions can have dimension exponentially higher than the input data.

For illustration, consider the influence maximization problem for modeling viral marketing and social influence cascading introduced by Domingos and Richardson [58, 113] and Kempe, Kleinberg, and Tardos [93]. The two commonly studied models are the ***independent cascade*** **(IC)** and *linear threshold* models. The former assigns probabilities to the edges of a network to model a stepwise stochastic influence. The latter captures stochastic cascading by a profile of n threshold functions, one for each node in the network. Thus, the size of an IC instance or a linear threshold instance is equal to the size of the underlying network.

Since influence maximization study the power of *group influence*, a natural mathematical model for group influence is an *influence spread function*, $\sigma : 2^V \to \mathbb{R}^+ \cup \{0\}$, represented as a set function over the ground set V [93]. For each $S \subseteq V$, $\sigma(S)$ measures the expected number of nodes that S can influence in the stochastic influence process. The dimensionality of the influence spread for the IC and linear threshold models is $O(n^2)$, because these influence instances are determined by $O(n^2)$ parameters. In contrast, the natural ambient dimensionality of σ is $2^n - 1$,[23] which is exponential in the size $n = |V|$ of the ground set!

The IC and linear threshold models are relatively special subfamilies of influence models. Both have submodular influence spread functions. General influence models can be specified by probability profiles of the form $\mathcal{I} = (V, \{p^{\mathcal{I}}_{S,T} : S \subseteq T \subseteq V\})$ satisfying for any S, $\sum_{T \supseteq S} p^{\mathcal{I}}_{S,T} = 1$, and two boundary cases: (1) $p^{\mathcal{I}}_{\emptyset,\emptyset} = 1$ and $p^{\mathcal{I}}_{\emptyset,T} = 0$, for all non-empty T; (2) $p^{\mathcal{I}}_{V,V} = 1$. The influence spread function becomes

$$\sigma_{\mathcal{I}}(S) = \mathbf{E}_{T \supseteq S}\left[p^{\mathcal{I}}_{S,T} \cdot |T|\right]. \tag{16}$$

Note that in this case, the dimensions of the general influence models are, in fact, much greater than the dimensions of the influence spread functions. So, mathematically, $\sigma_{\mathcal{I}}$ can be viewed as a succinct summary of the influence data $\mathcal{I} = (V, \{p_{S,T} : S \subseteq T \subseteq V\})$, which has dimension between 2^n and 2^{2n}, but higher than 2^n by an exponential factor. In fact, exponentially large models are natural in many applications. For example, in cooperative game theory, Shapley used a set function to define a coalitional game [116]. The game trees discussed in Section 1 and binary decision diagrams in Section 2 are both typically exponential in size; so are the truth tables (over hypercubes) of Boolean functions and hypergraphs in discrete mathematics.

5.2 Dimensionality Reduction and Heuristic Solutions

Dimensionality reduction implies succinct data summarization, and summarization typically implies loss of information. One needs to address a basic question regarding the fundamental trade-off between reducing the complexity and preserving the quality of data and models:

What is the right way to lose information in Big Models?

By nature, dimensionality reduction is an area rich with heuristics. Dimensionality reduction plays multiple roles in diverse settings across data sciences and network sciences. Thus, the best characterization of the quality of a reduction may depend on the role it plays. For example, in both theory and practice, dimensionality reduction can be used:

- for approximation and computational efficiency, and
- as solution concepts of complex modeling.

Below, we will discuss both. While our primary focus in this section will be on the latter, we will review some well-known methods and results of the former to establish a basis for comparative analyses. In each of the following two subsections, we structure the content into two parts: first, a list of examples, followed by a comprehensive discussion.

5.2.1 Dimensionality Reduction for Approximation and Computational Efficiency. Many computational tasks suffer from the curse of dimensionality [26]. Thus, like sampling in statistics, dimensionality reduction of data/models provides a practical tool to reduce computational complexity at the expense of accuracy. This role is more commonly played in today's data and network analyses. We will briefly highlight a few examples for comparative discussion in Section 5.2.2.

[23]Because $\sigma(\emptyset) = 0$ always holds, the dimension is $2^n - 1$ instead of 2^n.

- Cheeger-Fiedler Embedding of Graph Data: The first step of spectral partitioning can be viewed as a dimensionality reduction. It reduces the $\Theta(n^2)$-dimensional graph data $G = (V, E)$ to its n-dimensional Fiedler vector. Crucially, in the process, the spectral method also reduces the size of the search space (for partitions) from $\Theta(2^n)$ to n.
- Spectral Graph Embedding: Chan, Gilbert, and Teng [46] used a richer dimensionality reduction step that combined techniques from spectral partitioning and geometric partitioning methods. Beyond the Fiedler vector $\mathbf{v}_2$, the graph data $G = (V, E)$ is reduced to $\mathbf{v}_2, \ldots, \mathbf{v}_{d+1}$ for a fixed d, where $\mathbf{v}_i$ denotes the eigenvector associated with the ith smallest eigenvalue of the G's Laplacian. In other words, Chan et al. first defined a d-dimensional *spectral embedding* of G, by mapping $u \in V$ to the vector $(\mathbf{v}_2(u), \ldots, \mathbf{v}_{d+1}(u))$, and then applied the geometric partitioning method to this spectral embedding. The combined geometric spectral heuristic effectively expanded the search space of the traditional spectral method. As a result, better experimental results were obtained [46].
- Low-Rank Approximation of Matrix Data: Perhaps the most widely used dimensionality reduction method is the **singular value decomposition (SVD)**-based low-rank approximation of matrix data. Mathematically, every $m \times n$ real matrix $\mathbf{M}$ has a SVD into the sum of rank-one matrices defined by two sets of orthonormal vectors:

$$\mathbf{M} = \sum_{i=1}^{\min(m,n)} \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

  where $\sigma_1, \ldots \sigma_{\min(m,n)}$ are singular values sorted in descending order. Then, a sensible rank-k approximation of $\mathbf{M}$ is $M_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$.
- Random Projection of Geometric Data: For geometric data, the celebrated work of Johnson-Lindenstrauss [87] applies random projections to embed data into a lower-dimensional space.

For graph data, rather than directly reducing the dimensionality, several methods reduce the $O(n^2)$-dimensional graph data to $\tilde{O}(n)$-dimensional "manifolds" of simpler graph structures.

- Tree Embedding: For undirected graphs, trees are among the simplest family of connected graphs, and many optimization problems over trees can be solved in polynomial time. Alon, Karp, Peleg, and West [3] considered the question of whether it is possible to effectively embed every undirected graph using a distribution over spanning trees.
- Spectral Graph Sparsification: Sparse graphs are another family of "simpler" graphs. Inspired by preconditioning in numerical analysis, Spielman and Teng [124] addressed the question of whether it is possible to approximate the Laplacian of every weighted graph by a sparse Laplacian matrix.

Discussion: A common thread that runs through typical applications of these dimensionality reduction techniques is that the intended computational or optimization tasks could be too resource intensive when performed on the original data.[24] Thus, it is instrumental for computational efficiency to make data smaller and/or simpler. Meanwhile, both in theory and in practice, the objectives of computational tasks also provide well-defined theoretical measures for analyzing the effectiveness of dimensionality reduction in these applications.

For example, *how good is the spectral reduction of graph data for graph partitioning?* On the one hand, valuable information about graphs can be lost in this reduction, as demonstrated in the

[24]Dimensionality reduction has also effectively been used for denoising in machine learning, network clustering, and data mining.

counterexamples by Guattery, Miller, Alper, and Kahng [5, 77, 122]. On the other hand, Cheeger's inequality provides some condition-based assurances regarding the degree of information loss in the Cheeger-Fiedler embedding (in the context of graph partitioning).

How good is the SVD-based low-rank approximation of matrix data in data mining? The celebrated Eckart–Young theorem establishes that under the Frobenius norm, M_k is the best possible rank-k approximation to M, for all $k \in [\min(m,n)]$ [61]. Although minimizing the Frobenius norm in data loss may not guarantee optimal approximation for all applications, the Eckart–Young theorem—like Cheeger's inequality for graph partitioning—has often been used to provide a basic justification for heuristics using the SVD-based dimensionality reduction. Both are great theories in practice.

In some applications, dimensionality reduction techniques offer more direct guarantees for achieving their intended objectives. For example, the Johnson-Lindenstrauss theorem [87] showed that random projections are effective tools for reducing dimensionality while approximately preserving fundamental geometric measures, including distances and inner products. The theorem shows that in the approximation world, the intrinsic dimensionality of any n points is $O(\log n)$.

It is now well known that every graph can be approximated by a distribution of spanning trees [2, 3, 21, 63, 65] and every (weighted) Laplacian matrix can be approximated by a sparse Laplacian matrix [22, 23, 119, 124]. For the former, for every $G = (V, E)$, there is a distribution over its spanning trees, which is efficiently computable, that yields expected distortion at most $O(\log n \log\log n)$ [2, 3, 21, 63, 65]. Tree embeddings have also been extended to weighted graphs and discrete metrics [21, 65]. For sparsification, for every Laplacian matrix $\mathbf{L}$, there exists a Laplacian matrix $\tilde{\mathbf{L}}$ with $O(n)$ non-zero entries such that

$$(1-\epsilon)\cdot \mathbf{x}\mathbf{L}\mathbf{x}^T \le \mathbf{x}\tilde{\mathbf{L}}\mathbf{x}^T \le (1+\epsilon)\cdot \mathbf{x}\mathbf{L}\mathbf{x}^T. \tag{17}$$

When using the Johnson-Lindenstrauss embedding in algorithm design, an overall approximation guarantee can be achieved if (1) these basic geometric quantities are the defining parameters for the quality of solutions and (2) $O(\log n)$ dimensions are low enough for efficient algorithms. Similarly, several provably good scalable approximation algorithms are designed based on tree embedding and spectral graph sparsification [3, 120, 134].

5.2.2 Dimensionality Reduction as Solution Concepts. In contrast to the discussion above, characterizations of information loss in this role of dimensionality reduction are less numerical; they are often instead conceptual. Let us jump-start our discussion with a few classical examples.

- Social Choice: In 1951, Kenneth Arrow formulated his famous impossibility theorem in social choice theory, known today as *Arrow's impossibility theorem* [13, 14]. He studied the following voting problem: identifying a ranking order of n candidates that best represents the collective preferences of a group of m voters who cast rank-based preference votes.
  Mathematically, it aims for the best function $c : S_n^m \to S_n$ that summarizes any m permutations $\pi_1, \ldots, \pi_m$ from the symmetric group S_n (each representing individual preferences over candidates) into a single permutation in S_n (representing their collective preference). This collective preference function c can be viewed as a dimensionality reduction from $\Theta(mn)$ dimensions to n dimensions.
- Coalitional Games: In 1951, Lloyd Shapley introduced a fundamental solution concept for modeling bargaining power in coalitional games [116]. His mathematical formulation is the following: Define a coalitional game over n players $V = \{1, \ldots, n\}$ as a set function $\mathbf{v} : 2^V \to \mathbb{R}^+ \cup \{0\}$, where $\mathbf{v}(\emptyset) = 0$.
  For any $S \subseteq V$, $\mathbf{v}(S)$ represents the collective powers of the group. Shapley was looking for a function ϕ that maps the data of group power to individual bargaining power. He proposed

the following bargaining power function:

$$\phi_{\mathbf{v}}(i) = \mathbf{E}_{\pi \sim S_n}\left[\mathbf{v}(B_{\pi,i} \cup \{i\}) - \mathbf{v}(B_{\pi,i})\right], \tag{18}$$

where $B_{\pi,i}$ denotes set of players preceding player i in permutation π. In other words, formulation (18) measures player i's expected *marginal contribution* over the set preceding i. His solution concept, now known as the *Shapley value*, can be viewed as a dimensionality reduction from $\Theta(2^n)$ dimensions to n dimensions.

- Combinatorial Games: In the 1930s, Sprague [127] and Grundy [76] developed a mathematical theory for *impartial games*, the family of combinatorial games in which both players have the same options at every position in the game. Recall that a combinatorial game is specified by a rule set defining the set of game positions directly movable from each game position.
  Mathematically, for any position in a game, the most comprehensive information structure is the game tree rooted at that position. Sprague and Grundy discovered that the following integer function, now known as the *Grundy Value*, encodes important information about games. For a game position G, GrundyValue(G) can be recursively defined using the game tree of G by a local summarization process from leaves to the root: if there is no position for G to move to, then GrundyValue(G) = 0; otherwise, if $\{G_1, \ldots, G_t\}$ are next-step options for G, then,

$$\text{GrundyValue}(G) = mex(\{\text{GrundyValue}(G_1), \ldots, \text{GrundyValue}(G_t)\}, \tag{19}$$

  where for any integer set S, mex(S) returns the smallest value of $\mathbb{Z}^+ \cup \{0\} \setminus S$.
  One can view the Grundy value as a dimensionality reduction from game trees to integers.
- PageRank for Web Search: In 1996, Larry Page and Sergey Brin used random-walk-based Markov chains (see Equation (9)) to formulate PageRank and then applied it successfully for Web search. The PageRank algorithm can be viewed as a dimensionality reduction function from $\Theta(n^2)$-dimensional directed graphs to n-dimensional numerical rankings.

Discussion: Dimensionality reduction is at the heart of data and model summarization. Each of these four examples above aims to construct a *desirable way* to summarize complex high-dimensional data. The lack of quantitative objective functions for these data summarization tasks has inspired mathematicians in the field to take an axiomatic approach to characterize proposed solution concepts, to capture fundamental difficulties in dimensionality reduction, and to guide compromises in selecting between *imperfect solutions*. We now summarize the principles underlying these classical results. In Section 5.3, we will give an example in network sciences to illustrate their applications.

Arrow's impossibility theorem in social choice theory highlights the fundamental challenges in data summarization. It establishes that under the following three basic fairness conditions, there is no voting function for collective preference summarization (for three or more candidates).

(1) *Unanimity*: If all voters cast an identical preference, then that preference must be the collective preference.
(2) *Non-dictatorship*: The collective preference cannot just be the preference of a particular voter.
(3) ***Independence of Irrelevant Alternatives (IIA):*** If two elections have identical individual preferences between two candidates, then they have the same collective preference between these two candidates.

In other words, Arrow's theorem shows that dimensionality reduction for preference summarization must be in some sense "heuristic" in that certain desirable fairness conditions, e.g., IIA, may not be strictly satisfied.

Shapley's formulation is an impressive reduction in dimensionality. By going from 2^n-dimensions to n dimensions, it creates an aggressive exponential compression in data to generate its summarization.

At first glance, the reduction Equation (18) looks reasonable but heuristic. But Shapley developed a beautiful axiomatic theory to characterize it. First, he highlighted four basic desirable conditions satisfied by his formulation:

(1) *Efficiency*: $\forall \mathbf{v}, \sum_{i \in V} \phi_{\mathbf{v}}(i) = \mathbf{v}(V)$.
(2) *Symmetry*: $\phi_{\mathbf{v}}$ is permutation invariant.
(3) *Linearity*: $\forall \mathbf{v}, \mathbf{u}$ over V, and $\forall \alpha, \beta \in \mathbb{R}$, $\phi_{\alpha \cdot \mathbf{v} + \beta \cdot \mathbf{u}} = \alpha \cdot \phi_{\mathbf{v}} + \beta \cdot \phi_{\mathbf{u}}$.
(4) *Null Player*: For any i, if $\forall S \subseteq V \setminus \{i\}$, $\mathbf{v}(S \cup \{i\}) - \mathbf{v}(S) = 0$, then $\phi_{\mathbf{v}}(i) = 0$.

Remarkably, he proved the following statement: The reduction Equation (18) is the only reduction from n-variable set functions to n-dimensional vectors satisfying these four conditions.

Shapley's theorem provides a landmark example of how dimensionality-reduction-based solution concepts can and should be characterized. To me, Shapley's characterization is an illuminating theory. It captures the essence underlying a heuristic and provides a guiding principle for both network sciences and big data computing.

Sprague and Grundy defined a notion of *game value* to summarize game trees, to satisfy two basic mathematical conditions:

(1) *Outcome Revealing*: One can determine from the game value alone the game outcome (i.e., whether or not the current player has a winning strategy) without needing to evaluate the game tree.
(2) *Equivalence*: If two games have the same game value, then they are exchangeable in any game in which they are components.

In their work, Sprague and Grundy considered a special but popular notion of one game being a component game of another. For any two impartial games G and H, the *disjunctive sum* of the two, denoted by $(G + H)$, is a game in which, at each step, a player can move either in G or in H (but not both). Sprague-Grundy theory [35, 76, 127] establishes that

$$\text{GrundyValue}(G + H) = \text{GrundyValue}(G) \oplus \text{GrundyValue}(H) \quad \forall \text{impartial} \quad G, H, \tag{20}$$

where $\oplus$ is the bit-wise exclusive-or operator.

Furthermore, $\text{GrundyValue}(G) \neq 0$ if and only if the current player has a winning strategy. Thus, they proved that with respect to the disjunctive sum, a game tree summarization exists that satisfies both *Outcome Revealing* and *Equivalence* conditions.[25] It is worth noting that the Grundy-value-based equivalence applies only to disjunctive sums, and does not hold for other natural ways to combine combinatorial games.

PageRank is one of the most effective formulations of *network centrality* for Web search. It has been validated by large-scale empirical data in the real-world. Theoretically, Alon Altman and Moshe Tennenholtz [6] identified five basic conditions satisfied by the ranking order induced by the PageRank.

(1) *Symmetry*: the ranking order is permutation invariant.
(2) *Self Edge*: Adding a directed self-loop to a node v will not change the relative ranking order of any other pair of nodes, but it will make v strictly better than any node that was not strictly better than v before the self-loop was added.

[25]However, Grundy values can be intractable to compute even for polynomial-time solvable games [45].

(3) *Vote By Committee*: Locally replacing the star from a node v to its out-neighbors $N^{out}(v)$ by a star to a new set of nodes (delegated to a committee) and complete bipartite links from the committee to $N^{out}(v)$ will not change the ranking order of any pair in the original network.
(4) *Collapsing*: roughly speaking, a complete bipartite "committee" of nodes can be contracted into a single node without changing the ranking order of other nodes.
(5) *Proxy*: If a node v has the same number of in-neighbors as out-neighbors, and all its in-neighbors have identical ranking, then v and the links incident to v can be replaced by a perfect matching between $N^{in}(v)$ and $N^{out}(v)$ without changing the ranking order.

Altman and Tennenholtz proved that the *ordinal* ranking order defined by the numerical PageRank can be uniquely characterized by these five conditions.

These milestone frameworks have provided valuable guidance for understanding data summarization and dimensionality reduction in network sciences and data sciences. Arrow's impossibility theorem has encouraged generations of scholars to develop refined theory for modeling desirable properties and basic challenges in voting heuristics and formulations. Like the landmark Gödel's incompleteness theorem and Church-Turing's undecidability theory before it, Arrow's impossibility theorem distinctly characterizes a fundamental limitation. These classical results have motivated fundamental impossibility theorems to study broad classes of problems, including clustering [96], fairness [97, 98], and learnability [27].

Shapley's intuitive formulation and concise characterization of bargaining power have inspired many to seek concise and yet precise axioms for modeling solution concepts in summarizing multifaceted data. Sprague-Grundy theory provides a foundational example for characterizing model summarization through a combination of axiomatic properties and subclass conditions. This seminal characterization of game trees for impartial games inspired subsequent seminal works on the values of games. Notably, Berlekamp, Conway, and Guy's "Winning Ways for Your Mathematical Plays" [28] and Conway's "On Numbers and Games"[26] [53]. Collectively, these works laid the groundwork for **Combinatorial Game Theory (CGT)**.

By demonstrating convincingly the utility of PageRank in real-world Web Search, Brin and Page's work highlighted a fundamental empirical approach to capture dimensionality-reduction-based solution concepts for network sciences. The subsequent axiomatic characterization of Altman and Tennenholtz is an example of modeling the *property* of summarization, i.e., the ranking order induced by PageRank rather than PageRank values themselves.

5.3 An Axiomatic Framework for Modeling Influence Centrality

We conclude this section with an axiomatic characterization of a heuristic for dimensionality reduction in network influence models. Recall that in the general model, an influence instance is specified by a probability profile of the form $\mathcal{I} = (V, \{p^{\mathcal{I}}_{S,T} : S \subseteq T \subseteq V\})$ that satisfies $\forall S, \sum_{T \supseteq S} p^{\mathcal{I}}_{S,T} = 1$, and boundary conditions: $p^{\mathcal{I}}_{\emptyset,\emptyset} = 1$ and $p^{\mathcal{I}}_{\emptyset,T} = 0, \forall T \neq \emptyset$, and $p^{\mathcal{I}}_{V,V} = 1$. Recall also that the influence spread function of $\mathcal{I}$ is $\sigma_{\mathcal{I}}(S) = \mathrm{E}_{T \supseteq S}[p^{\mathcal{I}}_{S,T} \cdot |T|]$.

The solution concept that we will discuss is *influence centrality*, which summarizes influence data. Formally, an influence centrality ψ is a map from influence instances to $|V|$-dimensional real vectors.

The *Shapley influence centrality* of $\mathcal{I}$ is defined as the Shapley value of its influence spread function $\sigma_{\mathcal{I}}$. Natural and intuitive, this centrality definition is the composition of two well-established dimensionality reductions, first from a space of dimension exponentially larger than $2^{|V|}$ to $\Theta(2^{|V|})$ dimensions, and then from $\Theta(2^{|V|})$ dimensions to $|V|$ dimensions.

[26]Also see Knuth's beautiful novel on surreal numbers [100].

We now ask two natural questions:

> *Is this a meaningful solution concept? How should we capture the meaning of this influence centrality?*

Following Shapley's framework, one can begin by writing down basic conditions satisfied by the Shapley influence centrality.

1. *Symmetry*: The centrality is permutation invariant.
2. *Normalization*: The centrality has a mean value equal to 1.
3. *Bayesian*: Before stating this condition, let us start with a definition. For any two influence instances $\mathcal{I}_1$ and $\mathcal{I}_2$ over a common ground set V, and a prior distribution $(\alpha, 1-\alpha)$ over the two, the *Bayesian influence instance*, denoted by $\alpha \cdot \mathcal{I}_1 + (1-\alpha) \cdot \mathcal{I}_2$ is defined as follows: For any *seed set* $S \subseteq V$, its influence is determined by randomly drawing $i \sim \{1,2\}$ with probabilities α and $(1-\alpha)$, respectively, and then applying the influence instance $\mathcal{I}_i$ to S.
   We now say an influence centrality ψ satisfies the *Bayesian* condition if $\psi_{\alpha \cdot \mathcal{I}_1 + (1-\alpha) \cdot \mathcal{I}_2} = \alpha\psi_{\mathcal{I}_1} + (1-\alpha)\psi_{\mathcal{I}_2}, \forall \mathcal{I}_1, \mathcal{I}_2, \alpha \in [0,1]$.
4. *Independence of Sink Nodes*: An influence instance may have some nodes, known as *sink nodes*, which have no influence on other nodes. Formally, $v \in V$ is a sink node in $\mathcal{I}$ if $\forall S, T \subseteq V \setminus \{v\}, p^{\mathcal{I}}_{S\cup\{v\}, T\cup\{v\}} = p^{\mathcal{I}}_{S,T} + p^{\mathcal{I}}_{S,T\cup\{v\}}$. Intuitively, because a sink node v has no influence on other nodes, either by itself or in a group, we should be able to "remove" it to obtain a projection of the influence instance. Let $\mathcal{I} \setminus \{v\}$ denote the influence instance over $V \setminus \{v\}$ such that for all $S, T \subseteq V \setminus \{v\}, p^{\mathcal{I}\setminus\{v\}}_{S,T} = p^{\mathcal{I}}_{S,T} + p^{\mathcal{I}}_{S,T\cup\{v\}}$.
   We say an influence centrality ψ satisfies *Independence of Sink Nodes* if $\forall\mathcal{I}$ and for any pair of sink nodes u, v, the centrality of u remains unchanged after v is projected out.

In light of Shapley's original axioms for bargaining power, these four conditions are the most prominent and notable ones for the Shapley influence centrality. The *Bayesian* condition expresses its degree of consistency among different influence instances. The *Independence of Sink Nodes* condition, like the *Vote by Committee*, *Collapsing*, and *Proxy* conditions from the PageRank axioms, expresses its consistency between influence instances across different ground sets.

Although these four conditions do not uniquely characterize the Shapley influence centrality, they establish a foundational framework for its axiomatic characterization. Our intuition was drawn from Myerson's insight in his proof of Shapley's axiomatic characterization. Mathematically, one can view each influence instance over V as a vector in Euclidean space with exponentially high dimension. The *Bayesian* condition then implies that influence centrality is an affine map from this Euclidean space to $\mathbb{R}^{|V|}$. Thus, the influence centrality across all influence models is uniquely determined by its values on a set of "basis" influence instances.

This structural observation guided us to characterize the centrality of the following family of simple deterministic influence instances. We call them *AND-models*. Imagine that we have a set R and a single node v not in R. Then, the AND logic "$v = \mathrm{AND}(R)$" defines the following simple influence instance over $R \cup \{v\}$: (1) R influences $R \cup \{v\}$ with probability 1; (2) For any $S \subseteq R \cup \{v\}$ and $S \neq R$, S can only influence S itself. We denote this influence instance by $\mathcal{I}_{(v=AND(R))}$. Let us compute v's Shapley influence centrality in $\mathcal{I}_{(v=AND(R))}$. In Shapley's random ordering, if v is the last node, then its marginal influence is 0, because R already influenced $R \cup \{v\}$; otherwise, v's marginal influence is 1. Moreover, the probability that v is last in the random ordering is equal to $\frac{1}{|R|+1}$. Thus, v's Shapley value is $1 - \frac{1}{|R|+1} = \frac{|R|}{|R|+1}$. Therefore, we can add this as the fifth condition satisfied by Shapley centrality.

5. *Bargaining with a Critical Set*: In $\mathcal{I}_{(v=AND(R))}$ over $R \cup \{v\}$, the centrality of v is $\frac{|R|}{|R|+1}$.

How natural is this condition? The following proposition shows that Shapley's formulation can also be captured by Nash's bargaining game between a player representing R and a player representing v.

PROPOSITION 5.1 (NASH BARGAINING OF SOCIAL INFLUENCE [49]). *Property 5 is consistent with Nash's bargaining solution between the critical set R and node v.*

PROOF. Player R's influence utility is $|R| + 1$ and player v's influence utility is 1. Because v is a sink node, it agrees that player R's contribution is at least $|R|$. But R thinks that v's contribution is 0, because R can influence v. Thus, the *threat point* of this bargaining game is $(|R|, 0)$. The *slack* is $(|R| + 1) - (|R| + 0) = 1$. While player v represents itself, player R represents a coalition of size $|R|$. In this coalition, members must cooperate to be effective, as the absence of any member will not influence v. The need to cooperate to bargain with player v weakens player R. The ratio of v's bargaining strength to that of R is therefore $1 : \frac{1}{|R|}$. Then, Nash's bargaining solution provides a fair division of the slack between the player v and the player R:

$$(x1, x2) \in \text{argmax}_{x_1 \geq r, x_2 \geq 0, x_1 + x_2 = |R|+1} (x_1 - |R|)^{1/|R|} \cdot x_2 \tag{21}$$

Nash's solution to this optimization problem is

$$(x_1, x_2) = \left(|R| + \frac{1}{|R| + 1}, \frac{|R|}{|R| + 1} \right). \tag{22}$$

□

While basic, this family of AND-logic-based influence instances is, in fact, sufficiently expressive to uniquely characterize influence centrality.

THEOREM 5.2 (AXIOMATIC CHARACTERIZATION OF SHAPLEY INFLUENCE CENTRALITY [49]). *The Shapley influence centrality is the unique solution satisfying Conditions 1–5. Moreover, each of these five conditions is independent of the other four.*

PROOF. (Sketch) For a ground set V, we embed AND-logic-based influence instances as follows. For any $\emptyset \subset R \subseteq U \subseteq V$, let $\mathcal{I}_{R,U}$ be the following influence instance over V: $\forall S \supseteq R$, $p^{\mathcal{I}_{R,U}}_{S,U \cup S} = 1$, and $\forall S \not\supseteq R$, $p^{\mathcal{I}_{R,U}}_{S,S} = 1$. In $\mathcal{I}_{R,U}$, R is called the *critical set*, and U is called the *target set*. In other words, we have embedded an AND-logic from R to each node in U. Any seed set containing R will activate all nodes in U, but if any node in R is missing, then the seed set can only influence itself. In $\mathcal{I}_{R,U}$ over V, by *Symmetry*, all nodes in R have the same influence centrality as all nodes in U and all nodes in $V \setminus (R \cup U)$.

Furthermore, all nodes in $V \setminus R$ are sink nodes. By projecting out U, we obtain the *NULL instance*, in which every node is a sink node. By *Symmetry* and *Normalization*, each has influence centrality equal to 1. Then, the *Independence of Sink Nodes* condition implies that every node in $V \setminus (R \cup U)$ has influence centrality equal to 1 in $\mathcal{I}_{R,U}$ over V. Let u be an arbitrary node in U. By projecting out all nodes in $V \setminus R \cup \{u\}$, we obtain the influence instance $\mathcal{I}_{(u=AND(R))}$. By Condition (5) and the *Independence of Sink Nodes* condition, u has influence centrality $\frac{|R|}{|R|+1}$ both in $\mathcal{I}_{(u=AND(R))}$ and in $\mathcal{I}_{R,U}$ over V. Then by *Normaliztion*, every node in R has influence centrality $1 + \frac{|U|}{|R|(|R|+1)}$.

Thus, under Conditions (1–5), the influence centrality of $\mathcal{I}_{R,U}$ over V is uniquely defined. The characterization of Shapley influence centrality is then established by the following structural theorem: For any V, influence instances in $\{\mathcal{I}_{R,U} \mid \forall \emptyset \subset R \subset U \subseteq V\}$ are linearly independent of each other. Moreover, together with the NULL instance—where every node in V is a sink node—they form a basis for the space of all influence instances over V [49].

Theorem 5.2 then follows directly from the fact that the influence centrality satisfying the *Bayesian* condition is an affine mapping. Because Conditions *Symmetry*, *Normalization*, *Independence of Sink Nodes*, and *Bargaining with a Critical Set* uniquely determine the centrality of a basis in the space containing the entire influence instances over V, the Shapley influence centrality is the unique solution to these five conditions. □

One can classify these five axioms into two groups:

- ***Principled Axioms***: *Symmetry*, *Normalization*, *Bayesian*, and *Independence of Sink Nodes.*
- ***Choice Axiom***: *Bargaining with a Critical Set.*

While the first four axioms are basic desirable conditions for influence centrality, the last one is a choice made by the Shapley influence centrality. The choice axiom provides a comprehensive comparison between two influence centralities that satisfy all the principled axioms [49].

Note that the AND-logic-based instances cannot be realized by graph-theoretical influence models whose influence spread functions are submodular. In Reference [48], we showed that OR-logic can also be used to build "basis" instances for influence models.

In fact, this statement is part of a broader theorem [48], showing more general cascading influence profiles have a graph-theoretical basis. Specifically, this connection is made through the family of deterministic influence instances involving broadcasting over layered networks. Thus, under the principled axioms of *Symmetry* and *Bayesian*, several centralities based on graph theory can be naturally and uniquely extended to the complete set of stochastic influence models. Because the family of influence instances, rooted in broadcasting over networks, possesses a submodular influence spread function, this theorem further establishes that, under the principled axioms of *Symmetry* and *Bayesian*, influence centrality has a unique extension from submodular influence models to the entirety of influence models.

6 THINKING BEYOND TRADITIONAL FRAMEWORKS: SOME REMARKS

The design of effective heuristic methods and mathematical attempts to understand them help to bridge theory and practice in computing. While formal modeling of real-world problems and their impact on algorithmic performance can often be challenging, it can also be a fulfilling experience. Theory and practice can mutually motivate and benefit each other, a wisdom eloquently shared by Donald Knuth in 1989 [101].

> *"My experiences also strongly confirmed my previous opinion that the best theory is inspired by practice and the best practice is inspired by theory. The examples I've mentioned, and many others, convinced me that neither theory nor practice is healthy without the other. But I don't want to give the impression that theory and practice are just two sides of the same coin. No. They deserve to be mixed and blended, but sometimes they also need to be pure. I've spent many an hour looking at purely theoretical questions that go way beyond any practical application known to me other than sheer intellectual pleasure. And I've spent many an hour on purely practical things like pulling weeds in the garden or correcting typographic errors, not expecting those activities to improve my ability to discover significant theories. Still, I believe that most of the purely practical tasks I undertake do provide important nourishment and direction for my theoretical work; and I believe that the hours I spend contemplating the most abstract questions of pure mathematics do have a payoff in sharpening my ability to solve practical problems."*
>
> Donald E. Knuth, "Theory and Practice" (1989)

Knuth's perspective on the relationship between theory and practice was far-reaching at the time and profoundly impacted me as a young student and scholar in theoretical computer science. Today, in the age of Big Data, AI, and ubiquitous computing, his wisdom remains highly relevant and insightful. Compared to traditional problems in (theoretical) algorithm design and analysis, modern computational tasks, from web search to **large language models (LLMs)**, exhibit much fuzzier input-output behaviors and characterizations. Heuristic decisions are becoming essential, not only at the algorithmic level to enhance computational efficiency but also at the model/framework level to formulate meaningful solutions. Today, fundamental questions such as choosing the objective function for a data-mining problem [18, 19], determining the appropriate feature space for social network embeddings [75], selecting a diffusion-reaction equation for a generative model [118], or deciding on the regularization term for a learning task [16, 73] have all become domains in which heuristic design plays a pivotal role. These decisions impact both algorithms and computational solutions. Like the role of heuristics in Herbert Simon's formulation of bounded rationality [117], heuristic decisions in modeling data summarization, knowledge discovery, and learning generalization are themselves reflections of our understanding of data and solutions.

Machine learning is a field natural for beyond the worst-case analysis, with data and models rich in uncertainty, randomness, and approximation errors. Deep learning [73], deep generative frameworks [74], and LLMs [140] are also fertile grounds for heuristics. Compared to heuristics discussed earlier such as binary decision diagrams and spectral partitioning techniques, which were designed for specific domains of application, deep learning and LLMs are adaptive, general-purpose frameworks for computational solutions. While both utilize multilevel schemes, they also enhance the traditional multilevel framework used in numerical computing and combinatorial optimization in multiple fundamental aspects.

The traditional multilevel method discussed in Section 2.3 uses a pyramid-like hierarchical structure. This *stratified* hierarchy is built along a single scale-based or frequency-based dimension. The method utilizes a pair of projection and interpolation operators, along with gradient descent processes at each layer, to construct a consistent solution across the hierarchy [133]. The hierarchical structure provides an effective mechanism to integrate *local features*, largely captured by the layer-wise gradient descent processes, with *global features* captured by cross-layer transformations. In a more profound way, deep learning performs multiple data-driven projections and transformations between layers across its hierarchy. Essentially, it combines multiple adaptive multilevel processes, significantly enriching the integration of local and global features, making it a powerful scheme for mining high-dimensional data. Unlike typical deep learning architectures and traditional multi-level structures, large language models construct extensive *unstratified* n-gram hierarchies, which are sufficiently flexible to encode or tokenize concepts and language elements of varying granularity [40, 140]. These hierarchical representations are formed and embedded using advanced deep learning techniques trained on massive datasets. Enabling interactions with a large language model also requires rich multifaceted hierarchical analyses that incorporate multi-attention-based transformations [140].

Consequently, gaining a mathematical understanding of deep learning and large language models calls for the development of substantial new theories to analyze the intricacies of complex hierarchical processes. On a foundational level, our understanding can be deepened by studying the characterizations of *global* properties that can be captured and learned through efficient *local* processes across hierarchical architectures and, more generally, the power of multifaceted, data-driven hierarchical analysis.

It is worth noting that the characterization of local computation for global properties has long been a fundamental subject in traditional modeling and algorithms. For example, the mathematical theory of finite-element analysis can be seen as a characterization of numerical functions over a global geometric domain that can be accurately approximated by the sum of local elementary functions over simplicial elements [129]. The algorithm we discussed earlier for the significant PageRank approximation can also be viewed as a scheme to conduct local random-walk-based network analyses to compute the global network centrality [34]. The same can be said about the local reversed diffusion process used for global network influence maximization [33, 130, 131]. The mechanisms for combining local processes, however, are relatively simple in those earlier studies. In contrast, the hierarchical architectures of deep neural networks and training frameworks for large language models use far more sophisticated multilevel mechanisms for integrating local processes. Even in traditional settings, multilevel heuristics and processes are some of the most complex algorithms to analyze theoretically, especially when they involve randomness. But these basic settings have seen beautiful mathematical developments, including wavelet analysis [57] and spectral analysis [38, 120]. Recent results in the field of graph neural networks [109, 142], which focus on analyzing the expressive power of hierarchical operations in the reasoning of graph properties, highlight the great potential of these research directions.

Due to their versatility, learning through deep neural networks and interactive reasoning with large language models have also introduced novel algorithmic paradigms and *computational primitives* to theoretical computer science. In this theoretical field, *solvability* is traditionally captured by core computational classes, which are defined by self-contained verification processes. For instance, the traditional notion of efficient algorithms is characterized by polynomial-time solvability, and the verification process also produces a "*library*" of primitives. New polynomial-time algorithms can be built on a library of prior polynomial-time primitives—like network flows and linear programming—through polynomial-time Cook-Turing reductions. Once verified, new polynomial-time algorithms become part of this library of efficient primitives for future algorithm design. Similarly, new scalable algorithms are usually built on a library of prior scalable primitives like sorting, spanning trees, and Laplacian solvers; new NP-complete proofs are built on a library of prior NP-complete problems via Karp-reductions. These prior primitives are in the library not by assumption but by the fact that they come with mathematical proofs that are self-contained within the framework of computational complexity theory. Thus, these evolving libraries of primitives provide a continuing foundation for building algorithmic and complexity theory.

In contrast, deep learning technologies are powerful and *assumed primitives* for today's real-world computing and applications. Observing the nearly ubiquitous presence of AI technologies in the realm of computing, I have developed a profound appreciation for Berliner's postulation that "Intelligent heuristics are the future of computing." Intelligent heuristics have multidisciplinary applications from assisting medical diagnoses to economic decision making to astronomical discoveries. Recently, the neural-network-based learning framework—using machine learning techniques originated for developing AlphaGo—has discovered fast new algorithms for matrix multiplication, a fundamental algorithmic problem classically considered in theoretical computer science [66]. Machine learning technologies have also inspired several new data-driven frameworks for algorithm design, making algorithms more adaptive to practical instances [17, 78, 84]. Large language models have likewise started to play a role in facilitating discovery, interactive learning, and active learning in various disciplines.

From the perspective of traditional theoretical computer science, these *heuristic primitives* represent a novel type of algorithmic building block.

> *How should we go beyond our traditional theoretical framework to encapsulate the power and limitations of these heuristic primitives, to understand and characterize the scope of their implications for algorithm design and computing in general?*

One field to look to for inspiration is cryptography. Modern cryptography has achieved a fundamental breakthrough by establishing a theory built on a primitive set of assumptions, followed by self-contained verification processes. For example, many basic theorems state that "If a one-way function exists, then..." [32, 92]. What should be the "existence of a one-way function"-like hypotheses for artificial intelligence? Under what basic assumptions about these heuristic AI primitives can one approach automatic human-level conjecture generation and theorem proving? Under what basic assumptions can we prove meaningful impossibility results? What are the limitations of hierarchical numerical approximations from noisy datasets in symbolic reasoning?

ACKNOWLEDGMENTS

I first express my gratitude to ACM KDD 2022 General Chairs Aidong Zhang, and Huzefa Rangwala for inviting me to present my work as a keynote talk, which inspired this article. Special thanks to Huan Liu, Vipin Kumar, Yan Liu, Nitesh Chawla, Wei Chen, Ajitesh Srivastava, Chao Wang, Kyle Burke, Julian Asilis, Vatsal Sharan, Baining Guo, and Matthew Ferland for their invaluable input and inspiring discussions. I am also grateful to Irma Olmedo for proofreading the article and providing valuable feedback. Last, I thank Sharon Burks, Danny Sleator, Gary Miller, Margaret Reid-Miller, Wei-Min Shen, Yolanda Gil, Manpreet Khaira, and Craig Knoblock for sharing their memories of the CMU Computer Science Program from the 1980s.

REFERENCES

[1] Emmanuel Abbe, Enric Boix-Adsera, Matthew S. Brennan, Guy Bresler, and Dheeraj Nagaraj. 2021. The staircase property: How hierarchical structure can guide deep learning. In *Advances in Neural Information Processing Systems*, Vol. 34. 26989–27002.

[2] Ittai Abraham, Yair Bartal, and Ofer Neiman. 2008. Nearly tight low stretch spanning trees. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*. 781–790.

[3] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. 1995. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.* 24, 1 (Feb. 1995), 78–100.

[4] Noga Alon and Vitali D. Milman. 1985. λ_1, Isoperimetric inequalities for graphs, and superconcentrators. *J. Combinat. Theory* 38, Ser. B (1985), 73–88.

[5] Charles J. Alpert, Andrew B. Kahng, and So-Zen Yao. 1999. Spectral partitioning with multiple eigenvectors. *Discret. Appl. Math.* 90, 1-3 (1999), 3–26.

[6] Alon Altman and Moshe Tennenholtz. 2010. An axiomatic approach to personalized ranking systems. *J. ACM* 57, 4, Article 26 (May 2010), 26:1–26:35.

[7] Nina Amenta and Günter Ziegler. 1999. Deformed products and maximal shadows of polytopes. In *Number 223 in Contemporary Mathematics*, B. Chazelle, J. Goodman, and R. Pollack (Eds.). 57–90.

[8] Reid Andersen, Fan Chung, and Kevin Lang. 2007. Using PageRank to locally partition a graph. *Internet Math.* 4, 1 (2007), 1–128.

[9] Reid Andersen, Shayan Oveis Gharan, Yuval Peres, and Luca Trevisan. 2016. Almost optimal local graph clustering using evolving sets. *J. ACM* 63, 2, Article 15 (May 2016), 31 pages.

[10] Reid Andersen and Yuval Peres. 2009. Finding sparse cuts locally using evolving sets. In *Proceedings of the Annual ACM Symposium on Theory of Computing*. 235–244.

[11] Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. 2017. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*. ACM, 429–437.

[12] David Applegate, Guy Jacobson, and Daniel D. Sleator. 1991. *Computer Analysis of Sprouts*. Technical Report CMU-CS-91-144. Carnegie Mellon University Computer Science.

[13] Kenneth J. Arrow. 1950. A difficulty in the concept of social welfare. *J. Politic. Econ.* 58 (1950).

[14] Kenneth J. Arrow. 1963. *Social Choice and Individual Values* (2nd ed.). Wiley, New York.

[15] David Arthur, Bodo Manthey, and Heiko Röglin. 2011. Smoothed analysis of the K-means method. *J. ACM* 58, 5, Article 19 (Oct. 2011), 31 pages.

[16] Julian Asilis, Siddartha Devic, Shaddin Dughmi, Vatsal Sharan, and Shang-Hua Teng. 2023. Regularization and optimal multiclass learning. Retrieved from https://2309.13692

[17] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. 2018. Dispersion for data-driven algorithm design, online learning, and private optimization. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*. 603–614.

[18] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. 2009. Approximate clustering without the approximation. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*. 1068–1077.
[19] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. 2008. A discriminative framework for clustering via similarity functions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. 671–680.
[20] Stephen T. Barnard and Horst D. Simon. 1994. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurr. Pract. Exp.* 6, 2 (1994), 101–117.
[21] Yair Bartal. 1996. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS'96)*. 184.
[22] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. 2009. Twice-Ramanujan sparsifiers. In *Proceedings of the Annual ACM Symposium on Theory of Computing*. 255–262.
[23] Joshua D. Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: Theory and algorithms. *Commun. ACM* 56, 8 (Aug. 2013), 87–94.
[24] Rene Beier and Berthold Vöcking. 2004. Typical properties of winners and losers in discrete optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*. 343–352.
[25] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* 15, 6 (June 2003), 1373–1396.
[26] Richard Bellman. 1957. *Dynamic Programming*. Dover Publications.
[27] Shai Ben-David, Pavel Hrubes, Shay Moran, Amir Shpilka, and Amir Yehudayoff. 2019. Learnability can be undecidable. *Nat. Mach. Intell.* 1, 1 (2019), 44–48.
[28] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. 2001. *Winning Ways for your Mathematical Plays*. Vol. 1. A. K. Peters, Wellesley, MA.
[29] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. 2014. Smoothed analysis of tensor decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. 594–603.
[30] Aditya Bhaskara, Aidao Chen, Aidan Perreault, and Aravindan Vijayaraghavan. 2022. Smoothed analysis for tensor methods in unsupervised learning. *Math. Program.* 193, 2 (June 2022), 549–599.
[31] Hans L. Bodlaender. 1988. Dynamic programming on graphs with bounded treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*. 105–118.
[32] Dan Boneh and Victor Shoup. 2015. *A Graduate Course in Applied Cryptography*. Retrieved from https://crypto.stanford.edu/~dabo/cryptobook/draft_0_2.pdf
[33] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*. 946–957.
[34] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Shang-Hua Teng. 2014. Multi-scale matrix sampling and sublinear-time PageRank computation. *Internet Math.* 10, 1-2 (2014), 20–48.
[35] Charles L. Bouton. 1901. Nim, A game with a complete mathematical theory. *Ann. Math.* 3, 1/4 (1901), 35–39.
[36] James H. Bramble, Joseph E. Pasciak, and Jinchao Xu. 1990. Parallel multilevel preconditioners. *Math. Comp.* 55 (1990), 1–22.
[37] Achi Brandt. 1977. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.* 31, 138 (Apr. 1977), 333–390.
[38] William L. Briggs, Van Emden Henson, and Steve F. McCormick. 2000. *A Multigrid Tutorial: Second Edition*. Society for Industrial and Applied Mathematics.
[39] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw.* 30, 1-7 (1998), 107–117.
[40] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.* 29, 8–13 (Sep. 1997), 1157–1166.
[41] Alon Brutzkus, Amit Daniely, and Eran Malach. 2020. ID3 learns juntas for smoothed product distributions. In *Proceedings of the Conference on Learning Theory (COLT'20)*, Vol. 125. PMLR, 902–915.
[42] Randal E. Bryant. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.* 35, 8 (Aug. 1986), 677–691.
[43] Randal E. Bryant. 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.* 24, 3 (1992), 293–318.
[44] Thang Nguyen Bui and Curt Jones. 1993. A heuristic for reducing fill-in in sparse matrix factorization. In *Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing (PPSC'93)*. SIAM, 445–452.
[45] Kyle W. Burke, Matthew T. Ferland, and Shang-Hua Teng. 2021. Winning the war by (strategically) losing battles: Settling the complexity of Grundy values in undirected Geography. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS'21)*. IEEE, 1217–1228.
[46] Tony F. Chan, John R. Gilbert, and Shang-Hua Teng. 1995. *Geometric Spectral Partitioning*. Technical Report Tech. report CSL-94-15. Xerox PARC, Palo Alto, CA.

[47] Jeff Cheeger. 1970. A lower bound for the smallest eigenvalue of the Laplacian. In *Problems in Analysis*, R. C. Gunning (Ed.). Princeton University Press, 195–199.

[48] Wei Chen, Shang-Hua Teng, and Hanrui Zhang. 2020. A graph-theoretical basis of stochastic-cascading network influence: Characterizations of influence-based centrality. *Theor. Comput. Sci.* 824-825 (2020), 92–111.

[49] Wei Chen and Shang-Hua Teng. 2017. Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality. In *Proceedings of the 26th International Conference on World Wide Web*. 967–976.

[50] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player nash equilibria. *J. ACM* 56, 3, Article 14 (May 2009), 14:1–14:57.

[51] Fan R. K. Chung. 1996. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, Vol. 92. American Mathematical Society.

[52] Ken Clarkson, David Eppstein, Gary L. Miller, Carl Sturtivant, and Shang-Hua Teng. 1993. Approximating center points with and without linear programming. In *Proceedings of the 9th ACM Symposium on Computational Geometry*. 91–98.

[53] John H. Conway. 2000. *On Numbers and Games* (2nd ed.). A. K. Peters, Wellesley, MA.

[54] Amit Daniely, Nathan Srebro, and Gal Vardi. 2023. Efficiently learning neural networks: What assumptions may suffice? Retrieved from https://2302.07426

[55] George B. Dantzig. 1951. Maximization of linear function of variables subject to linear inequalities. In *Activity Analysis of Production and Allocation*, T. C. Koopmans (Ed.). 339–347.

[56] Ludwig Danzer, Branko Grünbaum, and Victor Klee. 1963. Helly's theorem and its relatives. In *Proceedings of the 7th Symposia in Pure Mathematics, American Mathematical Society*. 101–180.

[57] Ingrid Daubechies. 1992. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[58] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 57–66.

[59] W. E. Donath and Alan J. Hoffman. 1972. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Tech. Disclos. Bull.* 15 (1972), 938–944.

[60] W. E. Donath and Alan J. Hoffman. 1973. Lower bounds for the partitioning of graphs. *J. Res. Dev.* 17 (1973), 420–425.

[61] Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1, 3 (1936), 211–218.

[62] Herbert Edelsbrunner. 1987. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, NY.

[63] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. 2008. Lower-stretch spanning trees. *SIAM J. Comput.* 32, 2 (2008), 608–628.

[64] Shimon Even and Robert E. Tarjan. 1976. A combinatorial problem which is complete in polynomial space. *J. ACM* 23, 4 (Oct. 1976), 710–719.

[65] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. 2003. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*. 448–455.

[66] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* 610, 7930 (2022), 47–53.

[67] Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czech. Math. J.* 23, 2 (1973), 298–305.

[68] Miroslav Fiedler. 1975. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czech. Math. J.* 25, 100 (1975), 619–633.

[69] Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. 1980. Polynomial-time algorithms for permutation groups. In *Proceedings of the Annual Symposium on Foundations of Computer Science*. 36–41.

[70] Rong Ge, Qingqing Huang, and Sham M. Kakade. 2015. Learning mixtures of gaussians in high dimensions. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*. 761–770.

[71] John R. Gilbert, Gary L. Miller, and Shang-Hua Teng. 1998. Geometric mesh partitioning: Implementation and experiments. *SIAM J. Sci. Comput.* 19, 6 (Nov. 1998), 2091–2110.

[72] G. H. Golub and C. F. Van Loan. 1989. *Matrix Computations* (2nd ed.). Johns Hopkins Press.

[73] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

[74] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*. 2672–2680.

[75] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM, 855–864.

[76] Patrick M. Grundy. 1939. Mathematics and games. *Eureka* 2 (1939), 198—211.

[77] Stephen Guattery and Gary L. Miller. 1998. On the quality of spectral separators. *SIAM J. Matrix Anal. Appl.* 19, 3 (1998), 701–719.

[78] Rishi Gupta and Tim Roughgarden. 2020. Data-driven algorithm design. *Commun. ACM* 63, 6 (may 2020), 87–94.

[79] Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. 2020. Smoothed analysis of online and differentially private learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Article 772, 13 pages.

[80] Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. 2021. Smoothed analysis with adaptive adversaries. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science*. 942–953.

[81] Kenneth M. Hall. 1970. An r-dimensional quadratic placement algorithm. *Manage. Sci.* 17 (1970), 219–229.

[82] Taher H. Haveliwala. 2003. Topic-sensitive Pagerank: A context-sensitive ranking algorithm for web search. *Trans. Knowl. Data Eng.* 15, 4 (2003), 784–796.

[83] Bruce Hendrickson and Robert W. Leland. 1995. A multi-level algorithm for partitioning graphs. In *Proceedings of the Supercomputing Conference*, Sidney Karin (Ed.). 28.

[84] Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. 2014. Algorithm runtime prediction: Methods & evaluation. *Artif. Intell.* 206 (Jan. 2014), 79–111.

[85] Mark Jerrum and Alistair Sinclair. 1988. Conductance and the rapid mixing property for Markov chains: The approximation of permanent resolved. In *Proceedings of the Annual ACM Symposium on Theory of Computing*. 235–244.

[86] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. 1985. How easy is local search? In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (SFCS '85)*. 39–42.

[87] William B. Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.* 26, 1-1.1 (1984), 189–206.

[88] Adam Tauman Kalai, Alex Samorodnitsky, and Shang-Hua Teng. 2009. Learning and smoothed analysis. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*. 395–404.

[89] Sampath Kannan, Jamie Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. 2018. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 2231–2241.

[90] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* 20, 1 (Dec. 1998), 34.

[91] George Karypis and Vipin Kumar. 1999. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th Conference on Design Automation*. 343–348.

[92] Jonathan Katz and Yehuda Lindell. 2020. *Introduction to Modern Cryptography*, 2nd ed. Chapman & Hall/CRC.

[93] David Kempe, Jon Kleinberg, and Eva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 137–146.

[94] Brain W. Kernighan and Shen Lin. 1970. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* 49, 2 (1970).

[95] Victor Klee and George J. Minty. 1972. *How Good is the Simplex Algorithm?* Academic Press, 159–175.

[96] Jon Kleinberg. 2002. An impossibility theorem for clustering. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS'02)*. 463–470.

[97] Jon Kleinberg. 2018. Inherent trade-offs in algorithmic fairness. *SIGMETRICS Perform. Eval. Rev.* 46, 1 (June 2018), 40.

[98] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2017. Inherent trade-offs in the fair determination of risk scores. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS'17)*, Vol. 67. 43:1–43:23.

[99] Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5 (Sep. 1999), 604–632.

[100] D. E. Knuth. 1974. *Surreal Numbers: How Two Ex-students Turned on to Pure Mathematics and Found Total Happiness: A Mathematical Novelette*. Addison-Wesley Publishing Company. Retrieved from https://books.google.com/books?id=ZUkwDc3FokgC

[101] Donald E. Knuth. 1991. Theory and practice (keynote address for the 11th world computer congress, 1989). *Theor. Comput. Sci.* 90, 1 (Nov. 1991), 1–15.

[102] David Lichtenstein and Michael Sipser. 1980. GO is polynomial-space hard. *JACM* 27, 2 (1980), 393–401.

[103] Richard J. Lipton and Robert E. Tarjan. 1979. A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36 (1979), 177–189.

[104] László' Lovász and Miklós Simonovits. 1993. Random walks in a convex body and an improved volume algorithm. *Random Struct. Algor.* 4 (1993), 359–412.

[105] Gary L. Miller, Manpreet Khaira, and Thomas J. Sheffler. 1992. Nested dissection: A survey and comparison of various nested dissection algorithms. In *CMU Technical Report*.

[106] Gary L. Miller, Dafna Talmor, and Shang-Hua Teng. 1997. Optimal good-aspect-ratio coarsening for unstructured meshes. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*. 538–547.

[107] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. 1997. Separators for sphere-packings and nearest-neighbor graphs. *J. ACM* 44, 1 (Jan. 1997), 1–29.

[108] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. 1998. Geometric separators for finite-element meshes. *SIAM J. Sci. Comput.* 19, 2 (Mar. 1998), 364–386.

[109] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. In *Proceedings of the International Conference on Learning Representations.*

[110] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference.* 161–172.

[111] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. 1990. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* 11, 3 (May 1990), 430–452.

[112] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. 2011. Online learning: Stochastic, constrained, and smoothed adversaries. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11).* 1764–1772.

[113] Matthew Richardson and Pedro Domingos. 2002. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02).* 61–70.

[114] Thomas J. Schaefer. 1978. On the complexity of some two-person perfect-information games. *J. Comput. Syst. Sci.* 16, 2 (1978), 185–225.

[115] Alexander Schrijver. 1985. *The New Linear Programming Method of Karmarkar.* Technical Report September. Centrum Wiskunde & Informatica (CWI).

[116] Lloyd S. Shapley. 1971. Cores of convex games. *Int. J. Game Theory* 1, 1 (1971), 11–26.

[117] Herbert A. Simon. 1955. A behavioral model of rational choice. *Quart. J. Econ.* 69, 1 (1955), 99–118.

[118] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, 2256–2265.

[119] Daniel A. Spielman and Nikhil Srivastava. 2008. Graph sparsification by effective resistances. In *Proceedings of the 40th ACM Symposium on the Theory of Computing.* 563–568.

[120] Daniel A. Spielman and Shang-Hua Teng. 2004. Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Annual ACM Symposium on Theory of Computing.* 81–90.

[121] Daniel A. Spielman and Shang-Hua Teng. 2004. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM* 51, 3 (May 2004), 385–463.

[122] Daniel A. Spielman and Shang-Hua Teng. 2007. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra Appl.* 421, 2-3 (Mar. 2007), 284–305.

[123] Daniel A. Spielman and Shang-Hua Teng. 2009. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Commun. ACM* 52, 10 (Oct. 2009), 76–84.

[124] Daniel A. Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (July 2011), 981–1025.

[125] Daniel A. Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26.

[126] Daniel A. Spielman and Shang-Hua Teng. 2014. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Anal. Appl.* 35, 3 (2014), 835–885.

[127] Roland P. Sprague. 1935. Über mathematische Kampfspiele. *Tôhoku Math. J.* 41 (1935), 438—444.

[128] Ernst Steinitz. 1922. IIIAB12: Polyeder und Raumeinteilungen. *Encyclopädie der mathematischen Wissenschaften (in German)* Band 3 (Geometries) (1922), 1–139.

[129] Gilbert Strang and George Fix. 2008. *An Analysis of the Finite Element Method* (2nd ed.). Wiley.

[130] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 1539–1554.

[131] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 75–86.

[132] Shang-Hua Teng. 1991. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning.* Ph.D. Dissertation. Advisor: Gary Miller, Carnegie Mellon University, Pittsburgh, PA.

[133] Shang-Hua Teng. 1999. Coarsening, sampling, and smoothing: Elements of the multilevel method. In *Algorithms for Parallel Processing*, Michael T. Heath, Abhiram Ranade, and Robert S. Schreiber (Eds.). Springer, 247–276.

[134] Shang-Hua Teng. 2010. The Laplacian Paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation (TAMC'10).* Springer-Verlag, Berlin, 2–14.

[135] Shang-Hua Teng. 2016. Scalable algorithms for data and network analysis. *Found. Trends Theoret. Comput. Sci.* 12, 1-2 (2016), 1–261.

[136] Shang-Hua Teng. 2017. Network essence: PageRank completion and centrality-conforming Markov chains. In *A Journey Through Discrete Mathematics A Tribute to Jiří Matoušek, Martin Loebl, Jaroslav Nešetřil, and Robin Thomas.* Springer, New York, NY, 765–799.

[137] Helge Tverberg. 1966. A generalization of Radon's theorem. *J. London Math. Soc.* (1966), 123–128.
[138] Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY.
[139] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.* 16 (1971), 264–280.
[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates, 6000–6010.
[141] Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. 2010. Efficient clustering with limited distance information. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'10)*. 632–640.
[142] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*.

Received 15 August 2023; revised 10 October 2023; accepted 10 October 2023