

# Research Statement

Daniel R. Golovin

## Summary

My research in applied algorithms spans the areas of approximation algorithms, online algorithms, data structures, machine learning, and electronic commerce. My recent work has focused on two broad research agendas, namely *uniquely represented data structures*, and *optimization under uncertainty*.

**Uniquely Represented Data Structures.** An implementation of an abstract data type (ADT) on a machine model is uniquely represented if it encodes each ADT state with a unique machine state. For my Ph.D. thesis I developed the first efficient constructions of uniquely represented hash tables, linked lists, binary search trees, and many other data structures on the RAM model of computation [3, 4, 7]. These results reversed over thirty years of pessimism in the research literature [1, 5, 13, 18]. These data structures provide the foundation for *strongly history independent* systems that store *exactly* the information specified by their designs, such as a filesystem that can delete a file in such a way that there is provably no trace on the system that the file ever existed, at least at the level of the machine model (i.e., bits rather than magnetic moments). They also provide the basis for fast equality testing of complex objects which may prove useful for speeding up software verification; I intend to explore this possibility in future work.

**Optimization Under Uncertainty.** I have developed online algorithms [15, 16, 14] for some general classes of optimization problems, for example monotone submodular function maximization subject to a budget constraint, which are optimal in the *no-regret model*<sup>1</sup> assuming  $\mathbf{P} \neq \mathbf{NP}$ . These algorithms can be used to

- speed up satisfiability solvers, integer-linear program solvers, and theorem provers, as well as
- maximize ad revenue on online adwords auctions, and
- maximize the value obtained from conducting a limited number of experiments, with direct application to pollution detection, network monitoring, viral marketing, and finding the best blog posts to read.

Initial experiments show these algorithms have great potential in practice [17]. For example, a solver developed by my collaborator Matthew Streeter based on one of our algorithms recently *won awards in a major theorem prover competition* (see section 2).

Overall I enjoy working on theoretically deep problems that are well motivated by applications. I try to approach problems with a fresh perspective and identify new opportunities for attacking a problem of interest, often by discovering previously unsuspected connections with other research areas. My research style and foci often lead me to collaborations outside of theory, and at Carnegie Mellon I have been privileged to collaborate with experts in machine learning, security, networking, and operations research.

## 1 Uniquely Represented Data Structures

Consider the following *secure redaction* problem: given a classified document containing certain information deemed sensitive, create an unclassified version of it such that a powerful (i.e., computational unbounded) adversary cannot learn any sensitive information from the unclassified version.

This question is of some practical importance. To pick just one example from many, last year confidential technical data from McLaren and Ferrari was leaked by the Fédération Internationale de l'Automobile due

---

<sup>1</sup>In the no-regret model the goal of an online algorithm is to perform well in comparison to the best fixed solution in hindsight.

to faulty attempts at redaction<sup>2</sup>. Secure redaction is part of a more general problem. Computer users on a typical system leave significant clues to their recent activities, in the form of logs, unflushed buffers, files marked for deletion but not yet deleted, and so on. This can have significant security implications. To address these concerns, the notion of *history independent data structures* was devised. Roughly, a data structure is history independent if an adversary with access to the underlying hardware of the system (i.e., the memory layout of the data structure) can learn no more information than a legitimate user accessing the data structure via its standard interface.

The strongest possible guarantee of history independence is *unique representation* – that is, a data structure implementing some abstract data type (ADT) must encode each ADT state with a unique machine state. Such a canonical form ensures the current machine state can be inferred from the current ADT state, which allows the legitimate user to (information theoretically) simulate the adversary.

Unfortunately, thirty years of previous work had suggested that uniquely represented data structures are hopelessly inefficient relative to their conventional counterparts. There were several negative results [1, 5, 13, 18] showing that uniquely represented data structures for many fundamental ADTs were necessarily *exponentially* slower than their conventional counterparts in various models of computation. This includes queues, heaps, and ordered dictionaries. One bleak view of this state of affairs was provided by Lawrence Snyder, who wrote that his results “suggest a new general rule: for dynamic problems, avoid uniquely represented data structures” [13]. However, none of these negative results hold for randomized<sup>3</sup> data structures in the far more realistic RAM model, and it remained unknown whether any non-trivial data structures had efficient uniquely represented versions on actual machines. In my thesis work I proved that such data structures do exist [3, 4, 7]. In fact, not only is the overhead not exponential in this case – it is actually *small constants in both time and space* for almost all of the fundamental data structures I investigated, including arrays, stacks, queues, linked lists, hash tables, binary search trees, and heaps, among other things. One of the key insights leading to these results is the exploitation of a certain property of the Gale-Shapley stable matching algorithm [6] in the design of a uniquely represented hash table.

As discussed above, uniquely represented data structures store the absolute minimum amount of information necessary to be correct implementations. Here are some concrete examples of applications for which my work provides the first theoretically efficient solutions.

- **Filesystems** that have the property that deleting a file *provably leaves no trace whatsoever* that it ever existed, and reveals nothing about what order files were created or last modified or last accessed.
- **Voting Machines** that provably store only the set of participating voters, and nothing about the order in which they voted (which some voting protocols require be kept secret).
- **Databases** storing sensitive information (e.g., medical records) that provably reveal nothing about the order of record insertions or records that have been deleted, nor retain any evidence of previous queries.
- **Advanced File Formats** that maintain search indices or other data structures embedded in the file itself, yet provably store only the information specified by their creator via a well specified interface.

**Other Applications.** Parallel programs are notoriously hard to debug, in large part because execution order of the various threads of the program are not deterministic, but rather may change with each execution. Uniquely represented data structures can help, by significantly reducing the (possibly distributed) machine state’s dependence on the execution order of various threads. Uniquely represented data structures can also help speed up equality testing of complex data types, and might possibly be useful in deriving novel symmetry-breaking predicates for constraint satisfaction instances derived from software verification problems. Such predicates are often useful in speeding up the constraint solvers, and this is an interesting possibility that I intend to explore in future work.

<sup>2</sup><http://uk.eurosport.yahoo.com/22092007/13/fia-mistakenly-leaked-mclaren-ferrari-secrets.html>

<sup>3</sup>Randomization in this context means that the machine representation of each ADT state may depend on the random bits, but for any fixed sequence of random bits the data structure is uniquely represented.

## 2 No-Regret Algorithms for Online Optimization Problems

**Faster Solvers for Constraint Satisfaction Problems through Hybridization.** For many NP-hard problems, such as boolean satisfiability, integer-linear programming, and various constraint solving problems, there are many available solvers. These solvers are all known to take worst-case exponential time, but often do significantly better for some classes of inputs. Often different solvers perform well on different classes of inputs, which makes it possible in practice to combine different solvers into a “meta-solver” that outperforms each of its constituent parts. In this context each solver is treated as a “black-box”, and a combination of solvers is a *task-switching schedule* which specifies what solver to run in each time step.

Previous work mainly addressed the offline version of the problem (where a run-time distribution of each instance is known in advance for each solver), and either made unrealistic assumptions about the run-time distributions or took time exponential in the number of solvers. There were no results for the problem in its full generality, with arbitrary, unknown run-length distributions that might vary over time. Together with Matthew Streeter and Stephen Smith [17, 15, 16], I remedied this situation by providing an efficient online algorithm for the general case with extremely strong theoretical guarantees. One such guarantee is that for any  $\epsilon > 0$ , on any sufficiently long sequence  $\sigma$  of inputs this algorithm is guaranteed to take no more than  $4 + \epsilon$  times the execution time of the best fixed task-switching schedule on  $\sigma$ . Furthermore, we proved that the above factor of 4 is optimal for efficient online algorithms, assuming  $\mathbf{P} \neq \mathbf{NP}$ . Moreover, our algorithm maintains the above performance guarantee against the best task-switching schedule that can suspend and resume solvers, even when it is only allowed to terminate executing solvers. This is important in practice because constraint satisfaction solvers are typically memory bound.

Our results were obtained by the drawing on techniques for approximation algorithms and online algorithms in the no-regret model; we arranged an ensemble of no-regret algorithms for the multiarmed bandit problem to behave like a noisy version of a greedy approximation algorithm for the offline problem. In this case, the algorithm is reasonably simple in hindsight and easy to implement, however its performance guarantees often surprise researchers at first and its analysis is non-trivial. Moreover, our experiments (for SAT, planning, constraint solving, and theorem proving) show favorable speedups over the current best solvers, typically on the order of 20% to 300% depending on the problem domain [17, 15]. More recently, while I was finishing up my thesis Matthew Streeter used one of our algorithms to hybridize existing state-of-the-art SAT solvers and entered the resulting solver into a major theorem proving competition, CASC-J4<sup>4</sup>, where it came in first place in the two divisions that it was entered in.

**Online Maximization of Submodular Functions.** More recently, we have extended our work on task-switching schedules to the much more general framework in which the notion of a problem being solved is replaced by the value of an arbitrary monotone *submodular* function on the set of actions performed so far [14]. Submodularity is a natural diminishing returns property, which says the marginal benefit from performing an action  $a$  given the actions we have performed so far cannot increase as we perform other actions. We have also considered the *coverage objective*, where the goal is get as much work done (e.g., solve as many problems as possible), subject to a budget on the sequence of actions you can perform. We give algorithms whose performance ratios converge to the best possible constants (assuming  $\mathbf{P} \neq \mathbf{NP}$ ) *simultaneously* for both the coverage and minimum time to solution objectives. Our algorithms are applicable to the following problems.

- **Faster Optimization** by hybridizing solvers for hard computational problems.
- **Faster Databases.** Given a join operator with commutative filters, adaptively selecting which order to evaluate the filters can speed up the operation [2, 12]. Our algorithms provide a structured way to do this with significantly better theoretical guarantees than previous work.
- **Finding the Best Information Sources.** For example, for  $B \in \mathbb{N}$ , select the most informative set of  $B$  sources, be they news stories, blogs, podcasts, or something else. This problem makes use of the guarantees we have obtained for the coverage objective. Here, an action is an information source, and the “work” it does is the (user reported) value of the information the user gets from inspecting it.

---

<sup>4</sup>The 4<sup>th</sup> International Joint Conference on Automated Reasoning CADE ATP System Competition. The solver, named MetaProver 1.0, was entered in the SAT and FNT divisions. There are six divisions overall. For more information, see <http://www.cs.miami.edu/~tptp/CASC/J4/>. For full results, see <http://www2.cs.man.ac.uk/~tptp/CASC/J4/WWWFiles/DivisionSummary.html>

- **Better Online Advertising** by optimizing the list of ads displayed in response to web search queries.

I believe this last application is a particularly good source of interesting future research challenges, because online advertising involves multiple strategic parties. If we wish to deploy our algorithms into the existing *keyword auction* model, then very complex issues concerning *mechanism design* and market dynamics come into play. This leads to many rich open problems at the intersection of computer science and economics which I intend to explore.

### 3 Approximation Algorithms

In addition to the topics above, I have investigated several resource allocation problems. Since these problems are typically NP-hard, a common approach is to obtain an *approximation algorithm* for them, i.e., an efficient algorithm that outputs solutions that are guaranteed to be near optimal. In this area I have worked on algorithms for implementing quorum systems in networks to minimize congestion [11], algorithms for various problems that well approximate large classes of objective functions simultaneously [10], and two stage optimization problems in which one must make choices in an initial planning stage given a set of possible future scenarios and then respond to the realized scenario in a recourse stage [8, 9].

One problem I investigated in the two stage optimization model is that of designing approximation algorithms and mechanisms for markets with probabilistic supply and demand [8]. The basic motivation was that if there are many buyers with small, known probabilities of wanting an item (probabilistic demand), but who do not wish to commit to bidding for it, then we can use concentration of measure results (e.g., Chernoff bounds) to treat it like a deterministic demand. If the materialized demand exceeds what we have provisioned for, we simply buy back that demand at some cost. Note that this is exactly what airlines do when they overbook a flight and too many passengers show up. In [8] I give algorithms and truthful-in-expectation mechanisms for approximating social welfare for the far more general case in which each bidder wants a set of goods with some known probability and there is a combinatorial auction with probabilistic supply and demand. This paper represents an (admittedly modest) initial stab at a general question that interests me: *How can one design good mechanisms that incorporate contingency planning and probabilistic beliefs about the future in a principled manner?*

### 4 Future Research

In the future I intend to continue working on theoretically well-founded solutions to interesting problems of significant practical potential. I also plan on taking a more active role in applying the algorithms obtained to practical problems than I did as a graduate student. Below I discuss my plans to apply my recent theoretical results to various application domains, and my longer term plans to focus on resource allocation problems.

#### **Leverage theoretical work on uniquely represented data structures in various application domains.**

The theory of uniquely represented data structures is now sufficiently mature to warrant more applied work. I would like to work with researchers in the systems, privacy, security, and verification communities to explore the desirability of applying these data structures to, e.g., private-key databases, filesystems for flash thumb drives, file formats with secure redaction capabilities, and model checkers for software verification. I will also continue to investigate some remaining theoretical questions about uniquely represented data structures, such as the complexity of deterministic versions of them. Ongoing research suggests that such data structures may be closely related to certain error-correcting codes.

**Leverage my theoretical work on online optimization.** Our experiments with the algorithms in [15, 16, 14] demonstrate significant practical potential for these algorithms. I intend to investigate their use in a number of domains. For example, in a collaboration with Andreas Krause and Matthew Streeter, we are currently attempting to acquire the relevant data or simulation code to test our algorithms on problems in airport security (i.e., selecting distributions over guard patrols), selecting the  $k$  most informative blog posts for some fixed  $k$ , and environmental monitoring.

**Advance the state of the art in optimization, particularly online resource allocation algorithms.** In the long term, I intend to concentrate more on resource allocation problems. For example, in [14], we considered the problem of picking  $k$  actions from a set to maximize an unknown submodular reward function. The results we obtained give us hope that other extremely complex online problems might be possible to approximate well in the no-regret model. Interesting extensions to the model include:

- More general reward functions. In [15] we gave a positive result for combining heuristics for an interesting non-submodular reward function<sup>5</sup>. What is the most general class of reward functions  $f$  for which the online problem of picking  $k$  actions to maximize  $f$  can be well-approximated?
- More general constraints on action sets. In [15] any  $k$  actions are allowed. However, in general we might wish to deal with more complex constraints on the actions. For example, if each action consists of testing a water sample for pollution, and all actions are performed by a single boat, we might require the locations to be sampled to lie on a path that the boat can travel in a day. Alternately, some actions might have prerequisite actions, or some actions might be mutually exclusive.
- Incorporate (non-adversarial) reactive and strategic environments. In some settings, what actions you pick today might affect the reward function in the future. When and how can we deal with this setting effectively? Our current algorithms can deal effectively with adversarial environments, however if there is an opportunity to “cooperate” with the environment they might not exploit this. For example, if the algorithm happens to be playing iterated prisoners dilemma with a reasonable 2<sup>nd</sup> player but does not know this, it would be nice if the algorithm eventually converged on cooperative play anyway. An alternate example is allocating ad slots to ads in an adwords auction. An ideal online algorithm would not only guarantee high revenue with respect to the observed reward functions, but would also successfully encourage the environment to give it good reward functions by, for example, incentivising rational bidders to bid roughly their true valuation for a user click.

The ultimate question behind this line of research is: *Given a huge distributed system with largely unknown dynamics, what experiments and actions should an agent perform to maximize utility in the long term?* Researchers will have to address this question in order to fully exploit the growing opportunities created by ubiquitous computers and sensor networks. Within this field of inquiry, I am particularly interested in providing practical algorithms with strong theoretical guarantees for this problem – or more realistically for the important special cases of it that admit such algorithms. Even for the special cases I have considered so far, initial results suggest that such algorithms will produce significant benefits for society.

## References

- [1] Arne Andersson and Thomas Ottmann. New tight bounds on uniquely represented dictionaries. *SIAM Journal of Computing*, 24(5):1091–1103, 1995.
- [2] Shivnath Babu, Rajeev Motwani, Kamesh Munagala, Itaru Nishizawa, and Jennifer Widom. Adaptive ordering of pipelined stream filters. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of Data*, pages 407–418, 2004.
- [3] Guy E. Blelloch and Daniel Golovin. Strongly history-independent hashing with applications. In *FOCS '07: 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 272–282. IEEE, October 2007.
- [4] Guy E. Blelloch, Daniel Golovin, and Virginia Vassilevska. Uniquely represented data structures for computational geometry. In *SWAT '08: Proceedings of the 11th Scandinavian Workshop on Algorithm Theory*, pages 17–28, Gothenburg, Sweden, July 2008. Springer.
- [5] Niv Buchbinder and Erez Petrank. Lower and upper bounds on obtaining history independence. *Inf. Comput.*, 204(2):291–337, 2006.
- [6] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
- [7] Daniel Golovin. B-treaps: A uniquely represented alternative to B-trees. Manuscript.

---

<sup>5</sup>To handle the model where the schedule can suspend and resume processes rather than just terminating them.

- [8] Daniel Golovin. Stochastic packing-market planning. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 172–181, New York, NY, USA, 2007. ACM Press.
- [9] Daniel Golovin, Vineet Goyal, and R. Ravi. Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems. In B. Durand and W. Thomas, editors, *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science, STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 206–217. Springer-Verlag, 2006.
- [10] Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-Norms and All- $L_p$ -Norms approximation algorithms. In *FSTTCS '08: Proceedings of the 28th annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Bangalore, India, 2008. To appear. An earlier version appeared as Carnegie Mellon University technical report CMU-CS-07-153.
- [11] Daniel Golovin, Anupam Gupta, Bruce M. Maggs, Florian Oprea, and Michael K. Reiter. Quorum placement in networks: Minimizing network congestion. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 16–25, New York, NY, USA, 2006. ACM Press.
- [12] Kamesh Munagala, Shivnath Babu, Rajeev Motwani, Jennifer Widom, and Eiter Thomas. The pipelined set cover problem. In *International Conference on Database Theory*, pages 83–98, 2005.
- [13] Lawrence Snyder. On uniquely representable data structures. In *FOCS '77: IEEE Symposium on Foundations of Computer Science*, pages 142–146. IEEE, 1977.
- [14] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *NIPS: Advances in Neural Information Processing Systems 21*. MIT Press, Cambridge, MA, 2009. To appear. An earlier version appeared as Carnegie Mellon University technical report CMU-CS-07-171.
- [15] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Combining multiple heuristics online. In *AAAI '07: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1197–1203, Menlo Park, California, 2007. AAAI Press.
- [16] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Restart schedules for ensembles of problem instances. In *AAAI '07: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1204–1210, Menlo Park, California, 2007. AAAI Press.
- [17] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Combining multiple constraint solvers: Results on the CPAI'06 competition data. In *Proceedings of the Second International CSP Solver Competition*, pages 11–18, 2008.
- [18] Rajamani Sundar and Robert E. Tarjan. Unique binary search tree representations and equality-testing of sets and sequences. In *STOC '90: Proceedings of the twenty-second annual ACM Symposium on Theory of Computing*, pages 18–25, New York, NY, USA, 1990. ACM Press.