Science | DOI:10.1145/2949662 Marina Krakovsky

Reinforcement Renaissance

The power of deep neural networks has sparked renewed interest in reinforcement learning, with applications to games, robotics, and beyond.

Each time DeepMind has announced an amazing accomplishment in game-playing computers in recent months, people have taken notice.

First, the Google-owned, London-based artificial intelligence (AI) research center wowed the world with a computer program that had taught itself to play nearly 50 different 1980s-era Atari games—from Pong and Breakout to Pac-Man, Space Invaders, Boxing, and more—using as input nothing but pixel positions and game scores, performing at or above the human level in more than half these varied games. Then, this January, DeepMind researchers impressed experts with a feat in the realm of strategy games: AlphaGo, their Go-playing program, beat the European champion in the ancient board game, which poses a much tougher AI challenge than chess. Less than two months later, AlphaGo scored an even greater victory: it won 4 games in a best-of-5 series against the best Go player in the world, surprising the champion himself.

The idea that a computer can learn to play such complex games from scratch and achieve a proficient level elicits gee-whiz reactions from the general public, and DeepMind's triumphs have heightened academic and commercial interest in the AI field behind DeepMind's methods: a blend of deep neural networks and reinforcement learning called "deep reinforcement learning."

IMAGE BY MAXUSER

"Reinforcement learning is a model of learning where you're not given a solution—you have to discover it by trial and error," explains Sridhar Mahadevan, a professor at the University of Massachusetts Amherst, a longtime center of research into reinforcement learning.

The clearest contrast is with supervised learning, the kind used to train image recognition software, in which the supervision comes in the form of labeled examples (and requires people to label them). Reinforcement learning, on the other hand, "is a way of not needing labels, or labeling automatically by who's winning or losing—by the rewards," explains University of Alberta computer scientist Rich Sutton, a co-founder of the field of reinforcement learning and co-author of the standard textbook on the subject. In reinforcement learning, the better your moves are, the more rewards you get, "so you can learn to play the Go game by playing the moves and winning or losing, and no one has to tell you if that was a good move or a bad move because you can figure it out for yourself; it led to a win, so it was a good move."

Sutton knows the process is not as simple as that. Even in the neat, tightly controlled world of a game, deducing which moves lead to a win is a notoriously difficult problem because of the delay between an action and its reward, a key feature of reinforcement learning. In many games, you receive no feedback at all until the end of the game, such as a 1 for a win or a –1 for a loss.

"Typically, you have to go through hundreds of actions before your score increases," explains Pieter Abbeel, an associate professor at the University of California, Berkeley, who applies deep reinforcement learning to robotics. (For a robot, a reward comes for completing a task, such as correctly assembling two Lego pieces.) "Before you understand how the game works, and are learning through your own trial and error," Abbeel says, "you just kind of do things, and every now and then your score goes up, and every now and then it goes down, or doesn't go up. How do you tease apart which subset of things that you did contributed to your score going up, and which subset was just kind of a waste of time?"

This thorny question—known as the credit assignment problem—remains a major challenge in reinforcement learning. "Reinforcement learning is unique in that it's the only machine learning field that's focused on solving the credit assignment problem, which I think is at the core of intelligence," says computer scientist Itamar Arel, a professor of electrical engineering and computer sciences at the University of Tennessee and CEO of Osaro, a San Francisco-based AI startup. "If something good happens now, can I think back to the last *n* steps and figure out what I did that led to the positive or negative outcome?"

Despite the buzz around DeepMind, combining reinforcement learning with neural networks is not new.

Just as for human players, figuring out smart moves enables the program to repeat that move the next time it faces the same situation—or to try a new, possibly better move in hope of stumbling into an even higher reward. In extremely small worlds (think of a simple game like Tic-Tac-Toe, also known as Noughts and Crosses), the same exact situations come up again and again, so the learning agent can store the best action for every possible situation in a lookup table. In complex games like chess and Go, however, it is impossible to enumerate all possible situations. Even checkers has so much branching and depth that the game yields a mind-boggling number of different positions. So imagine what happens when you move from games to real-world interactions.

"You're never going to see the same situation a second time in the real world," says Abbeel, "so instead of a table, you need something that understands when situations are similar to situations you've seen before." This is where deep learning comes in, because understanding similarity—being able to extract general features from many specific examples—is the great strength of deep neural networks. These networks, whose multiple layers learn relevant features at increasingly higher levels of abstraction, are currently the best available way to measure similarities between situations, Abbeel explains.

The two types of learning—reinforcement learning and deep learning through deep neural networks—complement each other beautifully, says Sutton. "Deep learning is the greatest thing since sliced bread, but it quickly becomes limited by the data," he explains. "If we can use reinforcement learning to automatically generate data, even if the data is more weakly labeled than having humans go in and label everything, there can be much more of it because we can generate it automatically, so these two together really fit well."

Despite the buzz around DeepMind, combining reinforcement learning with neural networks is not new. TD-Gammon, a backgammon-playing program developed by IBM's Gerald Tesauro in 1992, was a neural network that learned to play backgammon through reinforcement learning (the TD in the name stands for Temporal-Difference learning, still a dominant algorithm in reinforcement learning). "Back then, computers were 10,000 times slower per dollar, which meant you couldn't have very deep networks because those are harder to train," says Jürgen Schmidhuber, a professor of artificial intelligence at the University of Lugano in Switzerland who is known for seminal contributions to both neural networks and reinforcement learning. "Deep reinforcement learning is just a buzzword for traditional reinforcement learning combined with deeper neural networks," he says.

Schmidhuber also notes the technique's successes, though impressive, have so far been in narrow domains, in which the current input (such as the board position in Go or the current screen in Atari) tells you everything you need to know to guide your next move. However, this "Markov property" does not normally hold outside of the world of games. "In the real world, you see just a tiny fraction of the world through your sensors," Schmidhuber points out, speaking of both robots and humans. As humans, we complement our limited perceptions through selective

memories of past observations; we also draw on decades of experience to combine existing knowledge and skills to solve new problems. "Our current reinforcement learners can do this in principle, but humans still do it much better," he says.

Researchers continue to push reinforcement learners' capabilities, and are already finding practical applications.

"Part of intelligence is knowing what to remember," says University of Michigan reinforcement learning expert Satinder Singh, who has used the world-building game Minecraft to test how machines can choose which details in their environment to look at, and how they can use those stored memories to behave better.

Singh and two colleagues recently co-founded Cogitai, a software company that aims to use deep reinforcement learning to "build machines that can learn from experience the way humans can learn from experience," Singh says. For example, devices like thermostats and refrigerators that are connected through the Internet of Things could continually get smarter and smarter by learning not only from their own past experiences, but also from the aggregate experiences of other connected devices, and as a result become increasingly better at taking the right action at the right time.

Osaro, Arel's software startup, also uses deep reinforcement learning, but promises to eliminate the costly initial part of the learning curve. For example, a computer learning to play the Atari game Pong from scratch starts out completely clueless, and therefore requires tens of thousands of plays to become proficient—whereas humans' experience with the physics of balls bouncing off walls and paddles makes Pong intuitive even to children.

"Deep reinforcement learning is a promising framework, but applying it from scratch is a bit problematic for real-world problems," Arel says. A factory assembling smartphones, for example, requires its robotic manufacturing equipment to get up to speed on a new design within days, not months. Osaro's solution is to show the learning agent what good performance looks like "so it gets a starting point far better than cluelessness," enabling the agent to rapidly improve its performance.

Even modest amounts of demonstration, Arel says, "give the agent a head start to make it practical to apply these ideas to robotics and other domains where acquiring experience is prohibitively expensive." C

Further Reading

Mnih, V., et al.
Human-level control through deep reinforcement learning. *Nature* (2015), vol. 518, pp. 529–533.

Silver, D., et al.
Mastering the game Go with deep neural networks and tree search. *Nature* (2016), vol. 529, pp. 484–489.

Tesauro, G.
Temporal difference learning and TD-Gammon. *Communications of the ACM* (1995), vol. 38, issue 3, pp. 58–68.

Sutton, R.S., and Barto, A.G.
Reinforcement Learning: An Introduction. MIT Press, 1998, https://mitpress.mit.edu/books/reinforcement-learning

Based in San Francisco, **Marina Krakovsky** is the author of *The Middleman Economy: How Brokers, Agents, Dealers, and Everyday Matchmakers Create Value and Profit* (Palgrave Macmillan, 2015).

Milestones

Computer Science Awards, Appointments

NEWEST AMERICAN ACADEMY MEMBERS INCLUDE SIX COMPUTER SCIENTISTS

Among the 176 new Fellows recently elected to the American Academy of Arts and Scientists are six in the computer science arena:

▸ Jeffrey A. Dean, Google. A Google Senior Fellow, Dean is a Fellow of ACM, and shared the ACM-Infosys Foundation Award for 2012 with Sanjay Ghemawat.

▸ Sanjay Ghemawat, Google. A research scientist who works with MapReduce and other large distributed systems, Ghemawat is also the author of the popular calendar application iCal, and with Dean, wrote the 2004 paper "MapReduce: Simplified Data Processing on Large Clusters."

▸ Anna R. Karlin, University of Washington. The Microsoft Professor of Computer Science & Engineering at the University of Washington, Karlin, an ACM Fellow since 2012, writes about the use of randomized packet markings to perform IP traceback, competitive analysis of multiprocessor cache coherence algorithms, unified algorithms for simultaneously managing all levels of the memory hierarchy, Web proxy servers, and hash tables with constant worst-case lookup time.

▸ Tom M. Mitchell, Carnegie Mellon University (CMU). Chair of the Machine Learning Department at CMU and E. Fredkin University Professor, Mitchell has contributed to the advancement of machine learning, artificial intelligence, and cognitive neuroscience.

▸ Tal D. Rabin, IBM T.J. Watson Research Center. Head of the cryptography research group at theWatson Research Center, Rabin's research focuses on the design of efficient, secure encryption algorithms, as well as secure distributed algorithms, the theoretical foundations of cryptography, number theory, and the theory of algorithms and distributed systems.

▸ Scott J. Shenker, University of California, Berkeley. Leader of the Initiatives Group and Chief Scientist of the International Computer Institute, Shenker received the 2002 SIGCOMM for lifetime contribution to the field of communication networks "For contributions towards an understanding of resource sharing on the Internet." He has been an ACM Fellow since 2003.

SCHNEIDER RECEIVES CRA SERVICE AWARD

The Computing Research Association (CRA) has named Fred Schneider, Samuel B. Eckert Professor and Chair of Computer Science at Cornell University, recipient of the Service to CRA Award for his ongoing work with the organization.

A member of the CRA Board from 2007 to 2016, Schneider served as chair of the organization's Government Affairs Committee for seven years, helping to drive CRA's policy agenda, and developing the Leadership in Science Policy Institute to educate computing researchers on how science policy in the U.S. is formulated.

Schneider led the organization's Committee on Best Practices for Hiring, Promotion, and Scholarship in conducting interviews with more than 75 academic and industry computing and information unit heads to understand issues and gain insights from practice. Preliminary recommendations were vetted with department chairs and CRA Deans at the CRA Conference at Snowbird in 2014, and were published in a CRA Best Practices memo, "Incentivizing Quality and Impact: Evaluating Scholarship in Hiring, Tenure, and Promotion."

The New York Times

TECHNOLOGY

Maybe We Should Leave That Up to the Computer

By DOUGLAS HEINGARTNER JULY 18, 2006

AMSTERDAM — Do you think your high-paid managers really know best? A Dutch sociology professor has doubts.

The professor, Chris Snijders of the Eindhoven University of Technology, has been studying the routine decisions that managers make, and is convinced that computer models, by and large, can do a better job of it. He even issued a challenge late last year to any company willing to pit its humans against his algorithms.

“As long as you have some history and some quantifiable data from past experiences,” Mr. Snijders claims, a simple formula will soon outperform a professional’s decision-making skills. “It’s not just pie in the sky,” he said. “I have the data to support this.”

Some of Mr. Snijders’s experiments from the last two years have looked at the results that purchasing managers at more than 300 organizations got when they placed orders for computer equipment and software. Computer models given the same tasks achieved better results in categories like timeliness of delivery, adherence to the budget and accuracy of specifications.

No company has directly taken Mr. Snijders up on his challenge. But a Dutch insurer, Interpolis, whose legal aid department has been expanding rapidly in

recent years, called in Mr. Snijders to evaluate a computer model it had designed to automate the routing of new cases — a job previously handled manually by the department's in-house legal staff.

The manager in charge of the project, Ludo Smulders, said the model was much faster and more accurate than the old system. "We're very satisfied about the results it's given our organization," he said. "That doesn't mean there are no daily problems, but the problems are much smaller than when the humans did it by hand. And it lets them concentrate more on giving legal advice, which is what their job is."

Mr. Snijders's work builds on something researchers have known for decades: that mathematical models generally make more accurate predictions than humans do. Studies have shown that models can better predict, for example, the success or failure of a business start-up, the likelihood of recidivism and parole violation, and future performance in graduate school.

They also trump humans at making various medical diagnoses, picking the winning dogs at the racetrack and competing in online auctions. Computer-based decision-making has also grown increasingly popular in credit scoring, the insurance industry and some corners of Wall Street.

The main reason for computers' edge is their consistency — or rather humans' inconsistency — in applying their knowledge.

"People have a misplaced faith in the power of judgment and expertise," said Greg Forsythe, a senior vice president at Schwab Equity Ratings, which uses computer models to evaluate stocks.

The algorithms behind so-called quant funds, he said, act with "much greater depth of data than the human mind can. They can encapsulate experience that managers may not have." And critically, models don't get emotional. "Unemotional is very important in the financial world," he said. "When money is involved, people get emotional." Many putative managerial qualities, like experience and intuition, may in fact be largely illusory. In Mr. Snijders's experiments, for example, not only do the machines generally do better than the managers, but some managers perform worse over time, as they develop bad habits that go uncorrected from lack

of feedback.

Other cherished decision aids, like meeting in person and poring over dossiers, are of equally dubious value when it comes to making more accurate choices, some studies have found, with face-to-face interviews actually degrading the quality of an eventual decision.

“People’s overconfidence in their ability to read someone in a half-an-hour interview is quite astounding,” said Michael A. Bishop, an associate professor of philosophy at Northern Illinois University who studies the social implications of these models.

And the effects can be serious. “Models will do much better in predicting violence than will parole officers, and in that case, not using them leads to a more dangerous society,” he said. “But people really don’t believe that the models are as accurate as they are.”

Models have other advantages beyond their accuracy and consistency. They allow an organization to codify and centralize its hard-won knowledge in a concrete and easily transferable form, so it stays put when the experts move on. Models also can teach newcomers, in part by explaining the individual steps that lead to a given choice. They are also faster than people, are immune to fatigue and give the human experts more time to work on other tasks beyond the current scope of machines.

So if they’re so good, why aren’t they already used everywhere?

Not everyone is convinced that managers are incorrigibly myopic. “I’ve never seen any evidence that there is a pattern of decline at all, and it just doesn’t fit with the way management literature is going, which is all around the emotional intelligence angle,” said Laura Empson, the director of the Clifford Chance Center of the Said Business School at Oxford University.

“I think there are a lot of people who have a strong technological orientation who would agree life would be a lot simpler if it weren’t for the humans,” she said. “But the reality is, organizations do have a lot of very intense and complicated human issues within them.”

Max H. Bazerman, a professor at Harvard Business School, wonders how

many managerial decisions can actually be modeled. “The vast majority of decisions that we make in professional life don’t have this quality,” he said.

He agrees that models can make better decisions about credit card applications and college admissions, he said, “but there are many decisions that are much more unique, where that database doesn’t exist. I’m as skeptical about human intuition as these folks, but it’s not only a computer model that we replace it with. Sometimes it’s thinking more clearly.”

Many in the field of computer-assisted decision-making still refer to the debacle of Long Term Capital Management, a highflying hedge fund that counted several Nobel laureates among its founders. Its algorithms initially mastered the obscure worlds of arbitrage and derivatives with remarkable skill, until the devaluation of the Russian ruble in 1998 sent the fund into a tailspin.

“As long as the underlying conditions were in order, the computer model was almost like a money machine,” said Roger A. Pielke Jr., a professor of environmental studies at the University of Colorado whose work focuses on the relation between science and decision-making. “But when the assumptions that went into the creation of those models were violated, it led to a huge loss of money, and the potential collapse of the global financial system.”

In such situations, “you can never hope to capture all of the contingencies or variables inside of a computer model,” he said. “Humans can make big mistakes also, but humans, unlike computer models, have the ability to recognize when something isn’t quite right.”

Another problem with the models is the issue of accountability. Mr. Forsythe of Schwab pointed out that “there’s no such thing as a 100 percent quantitative fund,” in part because someone has to be in charge if the unexpected happens. “If I’m making decisions,” he said, “I don’t want to give up control and say, ‘Sorry, the model told me.’ The client wants to know that somebody is behind the wheel.”

Still, some consider the continuing ascendance of models as inevitable, and recommend that people start figuring out the best way to adapt to the role reversal. Mark E. Nissen, a professor at the Naval Postgraduate School in Monterey, Calif., who has been studying computer-vs.-human procurement, sees a fundamental

shift under way, with humans becoming increasingly peripheral in making routine decisions, concentrating instead on designing ever-better models.

"The newest space, and the one that's most exciting, is where machines are actually in charge, but they have enough awareness to seek out people to help them when they get stuck," he said — for example, when making "particularly complex, novel, or risky" decisions.

The ideal future, then, may lie in letting computers and people each do what they do best. One way to facilitate this development is to train people to identify the typical cognitive foibles that lead to bad choices. "I've now worked with these models for so long," Mr. Snijders said, "that my instincts have changed along the way."

As Mr. Bishop of Northern Illinois University puts it, by making smart use of computer models' advantages, "you'll become like the crafty A student who doesn't work that hard but gets mostly right answers, rather than the really hard-working student who gets lots of wrong answers and as a result gets C's."

A version of this article appears in print on , on Page C4 of the New York edition with the headline: Maybe We Should Leave That Up to the Computer.

---

by

theguardian

Thu 1 Sep '16 06.00 BST

A few years ago, a young man named Kyle Behm took a leave from his studies at Vanderbilt University in Nashville, Tennessee. He was suffering from bipolar disorder and needed time to get treatment. A year and a half later, Kyle was healthy enough to return to his studies at a different university. Around that time, he learned from a friend about a part-time job. It was just a minimum-wage job at a Kroger supermarket, but it seemed like a sure thing. His friend, who was leaving the job, could vouch for him. For a high-achieving student like Kyle, the application looked like a formality.

But Kyle didn't get called in for an interview. When he inquired, his friend explained to him that he had been "red-lighted" by the personality test he'd taken when he applied for the job. The test was part of an employee selection program developed by Kronos, a workforce management company based outside Boston. When Kyle told his father, Roland, an attorney, what had happened, his father asked him what kind of questions had appeared on the test. Kyle said that they were very much like the "five factor model" test, which he'd been given at the hospital. That test grades people for extraversion, agreeableness, conscientiousness, neuroticism, and openness to ideas.

At first, losing one minimum-wage job because of a questionable test didn't seem like such a big deal. Roland Behm urged his son to apply elsewhere. But Kyle came back each time with the same news. The companies he was applying to were all using the same test, and he wasn't getting offers.

Roland Behm was bewildered. Questions about mental health appeared to be blackballing his son from the job market. He decided to look into it and soon learned that the use of personality tests for hiring was indeed widespread among large corporations. And yet he found very few legal challenges to this practice. As he explained to me, people who apply for a job and are red-lighted rarely learn that they were rejected because of their test results. Even when they do, they're not likely to contact a lawyer.

Behm went on to send notices to seven companies, including Home Depot and Walgreens, informing them of his intent to file a class-action suit alleging that the use of the exam during the job application process was unlawful. The suit, as I write this, is still pending. Arguments are likely to focus on whether the Kronos test can be considered a medical exam, the use of which in hiring is illegal under the Americans with Disabilities Act of 1990. If this turns out to be the case, the court will have to determine whether the hiring companies themselves are responsible for running afoul of the ADA, or if Kronos is.

But the questions raised by this case go far beyond which particular company may or may not

be responsible. Automatic systems based on complicated mathematical formulas, such as the one used to sift through Behm's job application, are becoming more common across the developed world. And given their scale and importance, combined with their secrecy, these algorithms have the potential to create an underclass of people who will find themselves increasingly and inexplicably shut out from normal life.

It didn't have to be this way. After the financial crash, it became clear that the housing crisis and the collapse of major financial institutions had been aided and abetted by mathematicians wielding magic formulas. If we had been clear-headed, we would have taken a step back at this point to figure out how we could prevent a similar catastrophe in the future. But instead, in the wake of the crisis, new mathematical techniques were hotter than ever, and expanding into still more domains. They churned 24/7 through petabytes of information, much of it scraped from social media or e-commerce websites. And increasingly they focused not on the movements of global financial markets but on human beings, on us. Mathematicians and statisticians were studying our desires, movements, and spending patterns. They were predicting our trustworthiness and calculating our potential as students, workers, lovers, criminals.

This was the big data economy, and it promised spectacular gains. A computer program could speed through thousands of résumés or loan applications in a second or two and sort them into neat lists, with the most promising candidates on top. This not only saved time but also was marketed as fair and objective. After all, it didn't involve prejudiced humans digging through reams of paper, just machines processing cold numbers. By 2010 or so, mathematics was asserting itself as never before in human affairs, and the public largely welcomed it.

Most of these algorithmic applications were created with good intentions. The goal was to replace subjective judgments with objective measurements in any number of fields - whether it was a way to locate the worst-performing teachers in a school or to estimate the chances that a prisoner would return to jail.

These algorithmic "solutions" are targeted at genuine problems. School principals cannot be relied upon to consistently flag problematic teachers, because those teachers are also often their friends. And judges are only human, and being human they have prejudices that prevent them from being entirely fair - their rulings have been shown to be harsher right before lunch, when they're hungry, for example - so it's a worthy goal to increase consistency, especially if you can rest assured that the newer system is also scientifically sound.

The difficulty is that last part. Few of the algorithms and scoring systems have been vetted with scientific rigour, and there are good reasons to suspect they wouldn't pass such tests. For instance, automated teacher assessments can vary widely from year to year, putting their accuracy in question. Tim Clifford, a New York City middle school English teacher of 26 years, got a 6 out of 100 in one year and a 96 the next, without changing his teaching style. Of course, if the scores didn't matter, that would be one thing, but sometimes the consequences are dire, leading to teachers being fired.

There are also reasons to worry about scoring criminal defendants rather than relying on a

judge's discretion. Consider the data pouring into the algorithms. In part, it comes from police interactions with the populace, which is known to be uneven, often race-based. The other kind of input, usually a questionnaire, is also troublesome. Some of them even ask defendants if their families have a history of being in trouble with the law, which would be unconstitutional if asked in open court but gets embedded in the defendant's score and labelled "objective".

It doesn't stop there. Algorithms are being used to determine how much we pay for insurance (more if your credit score is low, even if your driving record is clean), or what the terms of our loans will be, or what kind of political messaging we'll receive. There are algorithms that find out the weather forecast and only then decide on the work schedule of thousands of people, laying waste to their ability to plan for childcare and schooling, never mind a second job.

Their popularity relies on the notion they are objective, but the algorithms that power the data economy are based on choices made by fallible human beings. And, while some of them were made with good intentions, the algorithms encode human prejudice, misunderstanding, and bias into automatic systems that increasingly manage our lives. Like gods, these mathematical models are opaque, their workings invisible to all but the highest priests in their domain: mathematicians and computer scientists. Their verdicts, even when wrong or harmful, are beyond dispute or appeal. And they tend to punish the poor and the oppressed in our society, while making the rich richer. That's what Kyle Behm learned the hard way.

Finding work used to be largely a question of whom you knew. In fact, Kyle Behm was following the traditional route when he applied for work at Kroger. His friend had alerted him to the opening and put in a good word. For decades, that was how people got a foot in the door, whether at grocers, banks, or law firms. Candidates then usually faced an interview, where a manager would try to get a feel for them. All too often this translated into a single basic judgment: is this person like me (or others I get along with)? The result was a lack of opportunity for job seekers without a friend inside, especially if they came from a different race, ethnic group, or religion. Women also found themselves excluded by this insider game.

Companies like Kronos brought science into corporate human resources in part to make the process fairer. Founded in the 1970s by MIT graduates, Kronos's first product was a new kind of punch clock, one equipped with a microprocessor, which added up employees' hours and reported them automatically. This may sound banal, but it was the beginning of the electronic push - now blazing along at warp speed - to track and optimise a workforce.

As Kronos grew, it developed a broad range of software tools for workforce management, including a software program, Workforce Ready HR, that promised to eliminate "the guesswork" in hiring. According to its web page, Kronos "can help you screen, hire, and onboard candidates most likely to be productive - the best-fit employees who will perform better and stay on the job longer".

Kronos is part of a growing industry. The hiring business is becoming automated, and many of the new programs include personality tests like the one Kyle Behm took. It is now a $500 million annual business and is growing by 10 to 15% a year, according to Hogan Assessment

Systems Inc, a company that develops online personality tests. Such tests now are used on 60 to 70% of prospective workers in the US, and in the UK, according to the Association of Graduate Recruiters, 71% of employers use some form of psychometric test for recruitment.

Even putting aside the issues of fairness and legality, research suggests that personality tests are poor predictors of job performance. Frank Schmidt, a business professor at the University of Iowa, analysed a century of workplace productivity data to measure the predictive value of various selection processes. Personality tests ranked low on the scale – they were only one-third as predictive as cognitive exams, and also far below reference checks. "The primary purpose of the test," said Roland Behm, "is not to find the best employee. It's to exclude as many people as possible as cheaply as possible."

You might think that personality tests would be easy to game. If you go online to take a five factor personality test, it looks like a cinch. One question asks: "Have frequent mood swings?" It would probably be smart to answer "very inaccurate." Another asks: "Get mad easily?" Again, check no.

In fact, companies can get in trouble for screening out applicants on the basis of such questions. Regulators in Rhode Island found that CVS Pharmacy was illegally screening out applicants with mental illnesses when a personality test required respondents to agree or disagree with such statements as "People do a lot of things that make you angry" and "There's no use having close friends; they always let you down."

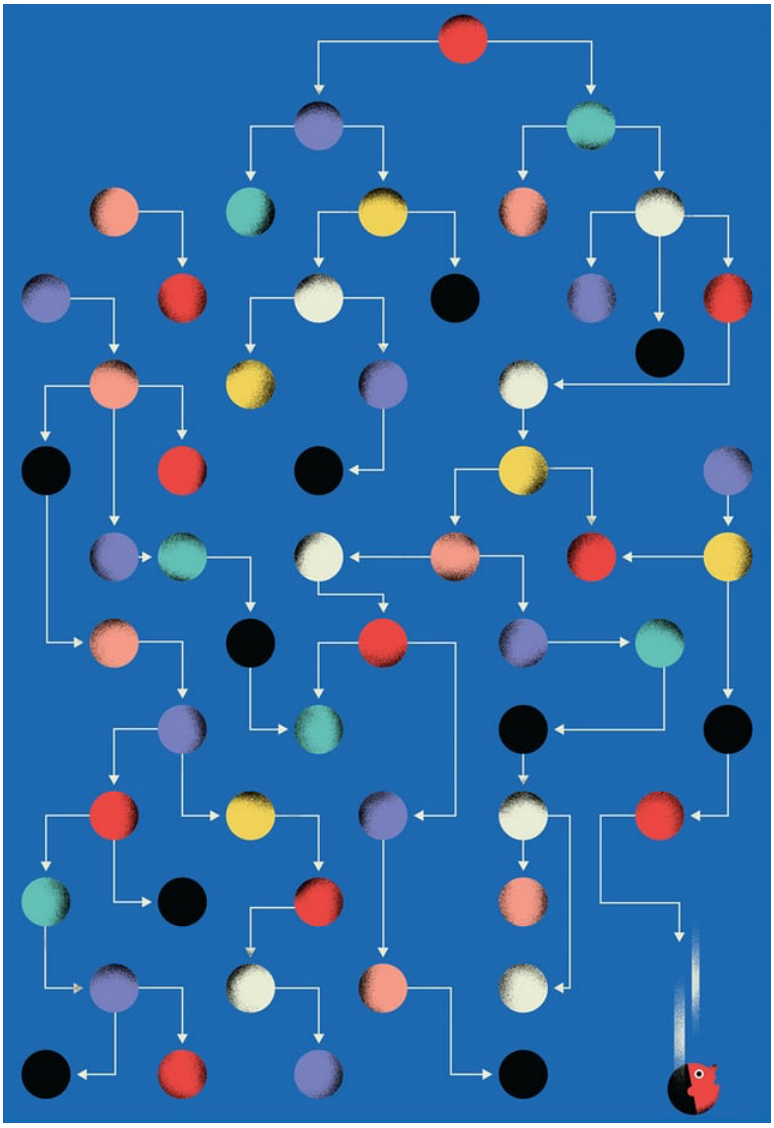

Illustration by Nathalie Lees

More intricate questions, which are harder to game, are more likely to keep the companies out of trouble. Consequently, many of the tests used today force applicants to make difficult choices, likely to leave them with a sinking feeling of “Damned if I do, damned if I don’t”.

McDonald’s, for example, recently asked prospective workers to choose which of the following best described them: “It is difficult to be cheerful when there are many problems to take care of” or “Sometimes, I need a push to get started on my work.”

In 2014, the Wall Street Journal asked a psychologist who studies behaviour in the workplace, Tomas Chamorro-Premuzic, to analyse thorny questions like these. The first of the two answers to the question from McDonald’s, Chamorro-Premuzic said, captured “individual differences in neuroticism and conscientiousness”; the second, “low ambition and drive”. So the prospective worker is pleading guilty to being either high-strung or lazy.

A Kroger supermarket question was far simpler: Which adjective best describes you at work, unique or orderly? Answering “unique”, said Chamorro-Premuzic, captures “high self-concept, openness and narcissism”, while “orderly” expresses conscientiousness and self-control.

Note that there's no option to answer "all of the above". Prospective workers must pick one option, without a clue as to how the program will interpret it. And some of the analysis will draw unflattering conclusions.

Defenders of the tests note that they feature lots of questions and that no single answer can disqualify an applicant. Certain patterns of answers, however, can and do disqualify them. And we do not know what those patterns are. We're not told what the tests are looking for. The process is entirely opaque.

What's worse, after the model is calibrated by technical experts, it receives precious little feedback. Sports provide a good contrast here. Most professional basketball teams employ data geeks, who run models that analyse players by a series of metrics, including foot speed, vertical leap, free-throw percentage, and a host of other variables. Teams rely on these models when deciding whether or not to recruit players. But if, say, the Los Angeles Lakers decide to pass on a player because his stats suggest that he won't succeed, and then that player subsequently becomes a star, the Lakers can return to their model to see what they got wrong. Whatever the case, they can work to improve their model.

Now imagine that Kyle Behm, after getting red-lighted at Kroger, goes on to land a job at McDonald's. He turns into a stellar employee. He's managing the kitchen within four months and the entire franchise a year later. Will anyone at Kroger go back to the personality test and investigate how they could have got it so wrong?

Not a chance, I'd say. The difference is this: Basketball teams are managing individuals, each one potentially worth millions of dollars. Their analytics engines are crucial to their competitive advantage, and they are hungry for data. Without constant feedback, their systems grow outdated and dumb. The companies hiring minimum-wage workers, by contrast, act as if they are managing herds. They slash expenses by replacing human resources professionals with machines, and those machines filter large populations into more manageable groups. Unless something goes haywire in the workforce - an outbreak of kleptomania, say, or plummeting productivity - the company has little reason to tweak the filtering model. It's doing its job - even if it misses out on potential stars. The company may be satisfied with the status quo, but the victims of its automatic systems suffer.

The majority of job applicants, thankfully, are not blackballed by automatic systems. But they still face the challenge of moving their application to the top of the pile and landing an interview. This has long been a problem for racial and ethnic minorities, as well as women.

The ideal way to circumvent such prejudice is to consider applicants blindly. Orchestras, which had long been dominated by men, famously started in the 1970s to hold auditions with the musician hidden behind a sheet. Connections, reputations, race or alma mater no longer mattered. The music from behind the sheet spoke for itself. Since then, the percentage of women playing in major orchestras has leapt by a factor of five - though they still make up only a quarter of the musicians.

The trouble is that few professions can engineer such an evenhanded tryout for job applicants.

Musicians behind the sheet can actually perform the job they're applying for, whether it's a Dvořák cello concerto or bossa nova on guitar. In other professions, employers have to hunt through CVs, looking for qualities that might predict success.

As you might expect, human resources departments rely on automatic systems to winnow down piles of résumés. In fact, in the US, some 72% of CVs are never seen by human eyes. Computer programs flip through them, pulling out the skills and experiences that the employer is looking for. Then they score each CV as a match for the job opening. It's up to the people in the human resources department to decide where the cutoff is, but the more candidates they can eliminate with this first screening, the fewer human hours they'll have to spend processing the top matches.

So job applicants must craft their résumés with that automatic reader in mind. It's important, for example, to sprinkle the résumé liberally with words the specific job opening is looking for. This could include previous positions (sales manager, software architect), languages (Mandarin, Java), or honours (summa cum laude). Those with the latest information learn what machines appreciate and what tangles them up, and tailor their applications accordingly.

The result of these programs is that those with the money and resources to prepare their résumés come out on top. Those who don't take these steps may never know that they're sending their résumés into a black hole. It's one more example in which the wealthy and informed get the edge and the poor are more likely to lose out.

So far, we've been looking at models that filter out job candidates. For most companies, those models are designed to cut administrative costs and to reduce the risk of bad hires (or ones that might require more training). The objective of the filters, in short, is to save money.

HR departments, of course, are also eager to save money through the hiring choices they make. One of the biggest expenses for a company is workforce turnover, commonly called churn. Replacing a worker earning $50,000 a year costs a company about $10,000, or 20% of that worker's yearly pay, according to the Center for American Progress. Replacing a high-level employee can cost as much as two years of salary.

Naturally, many hiring models attempt to calculate the likelihood that a job candidate will stick around. Evolv, Inc, now a part of Cornerstone OnDemand, helped Xerox scout out prospects for its call centres, which employ more than 40,000 people. The churn model took into account some of the metrics you might expect, including the average time people stuck around on previous jobs. But they also found some intriguing correlations. People the system classified as "creative types" tended to stay longer at the job, while those who scored high on "inquisitiveness" were more likely to set their questioning minds towards other opportunities.

But the most problematic correlation had to do with geography. Job applicants who lived farther from the job were more likely to churn. This makes sense: long commutes are a pain. But Xerox managers noticed another correlation. Many of the people suffering those long commutes were coming from poor neighbourhoods. So Xerox, to its credit, removed that highly correlated churn data from its model. The company sacrificed a bit of efficiency for

fairness.

While churn analysis focuses on the candidates most likely to fail, the more strategically vital job for HR departments is to locate future stars, the people whose intelligence, inventiveness, and drive can change the course of an entire enterprise. In the higher echelons of the economy, companies are on the hunt for employees who think creatively and work well in teams. So the modellers' challenge is to pinpoint, in the vast world of big data, the bits of information that correlate with originality and social skills.

A pioneer in this field is Gild, a San Francisco-based startup. Extending far beyond a prospect's alma mater or résumé, Gild sorts through millions of job sites, analysing what it calls each person's "social data". The company develops profiles of job candidates for its customers, mostly tech companies, keeping them up to date as the candidates add new skills. Gild claims that it can even predict when a star employee is likely to change jobs and can alert its customer companies when it's the right time to make an offer.

But Gild's model attempts to quantify and also *qualify* each worker's "social capital". How integral is this person to the community of fellow programmers? Do they share and contribute code? Say a Brazilian coder - Pedro, let's call him - lives in São Paulo and spends every evening from dinner to one in the morning in communion with fellow coders the world over, solving cloud-computing problems or brainstorming gaming algorithms on sites such as GitHub or Stack Overflow. The model could attempt to gauge Pedro's passion (which probably gets a high score) and his level of engagement with others. It would also evaluate the skill and social importance of his contacts. Those with larger followings would count for more. If his principal online contact happened to be Google's Sergey Brin, say, Pedro's social score would no doubt shoot through the roof.

But models like Gild's rarely receive such explicit signals from the data. So they cast a wider net, in search of correlations to workplace stardom wherever they can find them. And with more than six million coders in their database, the company can find all kinds of patterns. Vivienne Ming, Gild's chief scientist, said in an interview with Atlantic Monthly that Gild had found a bevy of talent frequenting a certain Japanese manga site. If Pedro spends time at that comic-book site, of course, it doesn't predict superstardom. But it does nudge up his score.

That makes sense for Pedro. But certain workers might be doing something else offline, which even the most sophisticated algorithm couldn't infer - at least not today. They might be taking care of children, for example, or perhaps attending a book group. The fact that prospects don't spend six hours discussing manga every evening shouldn't be counted against them. And if, like most of techdom, that manga site is dominated by males and has a sexist tone, a good number of the women in the industry will probably avoid it.

Despite these issues, Gild's category of predictive model has more to do with rewarding people than punishing them. It is tame compared with widely-used personality tests that exclude people from opportunities. Still, it's important to note that these hiring models are ever-evolving. The world of data continues to expand, with each of us producing ever-growing streams of updates about our lives. All of this data will feed our potential employers insights into us.

Will those insights be tested, or simply used to justify the status quo and reinforce prejudices? When I consider the sloppy and self-serving ways that companies often use data, I'm reminded of phrenology, a pseudoscience that was briefly popular in the 19th century. Phrenologists would run their fingers over the patient's skull, probing for bumps and indentations. Each one, they thought, was linked to personality traits. If a patient was morbidly anxious or suffering from alcoholism, the skull probe would usually find bumps and dips that correlated with that observation - which, in turn, bolstered faith in the science of phrenology.

Phrenology was a model that relied on pseudoscientific nonsense to make authoritative pronouncements, and for decades it went untested. Big data can fall into the same trap. Models like the ones that red-lighted Kyle Behm continue to lock people out, even when the "science" inside them is little more than a bundle of untested assumptions. •

Mail illustration: Nathalie Lees

This essay is adapted from Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy, published by Allen Lane on 6 September

• Follow the Long Read on Twitter at @gdnlongread, or sign up to the long read weekly email here

Topics

- The long read
- Psychology

- features

AI
ng

ictory

World Chess Champion Garry Kasparov (L) makes a move during his fourth game against the IBM Deep Blue chess computer. Credit: Stan Honda *Getty Images*

Twenty years ago IBM's Deep Blue computer stunned the world by becoming the first machine to beat a reigning world chess champion in a six-game match. The supercomputer's success against an incredulous Garry Kasparov sparked controversy over how a machine had managed to outmaneuver a grand master, and incited accusations—by Kasparov and others—that the company had cheated its way to victory. The reality of what transpired in the months and years leading up to that fateful match in May 1997, however, was actually more evolutionary than revolutionary—a Rocky Balboa–like rise filled with intellectual sparring matches, painstaking progress and a defeat in Philadelphia that ultimately set the stage for a triumphant rematch.

Computer scientists had for decades viewed chess as a meter stick for artificial intelligence. Chess-playing calculators emerged in the late 1970s but it would be another decade before a team of Carnegie Mellon University graduate students built the first computer—called Deep Thought—to beat a grand master in a regular tournament game. This success was short-lived—later that same year, 1989, Kasparov beat Deep Thought handily in the two games. IBM was impressed enough with the C.M.U. team's technology to bring its researchers onboard to develop an early version of Deep Blue—Deep Thought's successor. The Deep Blue team lost again to Kasparov in 1996 at a tournament in Philadelphia but managed to win one game out of six against the world champ.

That seemingly small victory "was very important to us to show that we were on the right track," says Deep Blue AI expert Murray Campbell, now a distinguished research staff member in the AI Foundations group within IBM T. J. Watson Research Center's Cognitive Computing organization. "By the time of our final match in 1997, we had made enough improvements to the system based on our experience that we were able to win." *Scientific American* spoke with Campbell about computer scientists' long obsession with chess, how IBM was able to turn the tables on the reigning chess champ and the challenges that lie ahead for AI.

[*An edited transcript of the interview follows.*]

How did you first get involved in the Deep Blue project?
I was part of a group of graduate students at Carnegie Mellon University that IBM approached. I had had a long interest in computer chess and had even written a chess program as an undergraduate. At C.M.U. I was working on artificial intelligence more generally and not exactly on building a high-performance chess computer that could play against a world champion. But as a side project a number of us [including Feng-hsiung Hsu and Thomas Anantharaman] did develop the machine that became known as Deep Thought, which became the first program to defeat a grand master, a professional level player in a tournament.

IBM noticed the successes that we were having building this machine on a shoestring budget and thought it would be interesting to have a group of us join IBM Research [in late 1989] to develop the next generation of this machine, called Deep Blue. They wanted to know if there was something special about the very best chess players in the world that was beyond what computers were capable of for the foreseeable future. Our feeling was that it was within a few years of being done, although other researchers thought it was still decades away.

What is it about chess that makes an especially interesting problem for a computer scientist?
Hundreds of millions of people around the world play chess. It's known as a game that requires strategy, foresight, logic—all sorts of qualities that make up human intelligence. So it makes sense to use chess as a measuring stick for the development of artificial intelligence.

When we look at a game like chess, we say, "Well, yes, of course computers do well because it's a well-defined game—the rules, the moves, the goals." And it's a constrained problem where you know all the information. Still, in spite of all those simplifications you could say chess is an enormously complex game, and that's why it took us, as a field, 50 years of development to finally beat the world

champion.

What was your role specifically on the Deep Blue team?
I was the AI expert. AI was quite different in 1989 and early 1990. The dominant part in those days was what we now called good old-fashioned AI, or symbolic AI, which was based less on machine learning. Certainly machine learning was a serious field in those days but nothing like what it is today, where we have massive data sets and large computers and very advanced algorithms to churn through the data and come up with models that can do some amazing things. When I started with IBM, machine learning methods for game-playing programs were fairly primitive and not able to help us much in building Deep Blue. We worked on algorithms for efficient search and evaluation of the possible continuations, which we knew Deep Blue would need in order to compete.

What were the most significant limitations on AI back then?
The hardware didn't really support building the kinds of large networks that have proven useful today in making big data models. And the data itself wasn't necessarily there to the extent that we needed it at that point. Any time you go back and look at the most popular computer systems from 20 or 25 years ago you're shocked at how you could get anything done on a system like that. But of course, we did—we didn't know what we were missing, I guess, because we had never experienced it.

As far as data, I don't think anybody had a clear idea back then that there was a big benefit. It wouldn't have paid to build a really large data set because in part the processing power wouldn't have been enough to use it anyway. So, we made do with much smaller data sets.

How useful was your own chess expertise in building Deep Blue?
Not as useful as you might think. I was able to, in the early stages, identify problems with the system and suggest approaches that I felt would be able to fix one problem without creating a host of other problems. That was probably good enough to get us to a certain point. Eventually, though, if you're going to be playing competitions there's a host of really game-specific knowledge you need to have. When we got closer to the point where we would actually be playing against a world champion we brought in grand masters —Joel Benjamin, in particular—to help us.

How did the grand masters help raise Deep Blue's game?
There were two parts to how they helped. One, in particular, was to help with the opening library, which every chess program uses in order to save time and make sure it gets into reasonable positions. Humans have been studying chess openings for centuries and developed their own favorite [moves]. The grand masters helped us choose a bunch of those to program into Deep Blue.

They also were, you could say, sparring partners for Deep Blue. They would play against the computer and try and pinpoint weaknesses of the system. And then we would sit around with them and with the rest of the Deep Blue team and try to articulate what that weakness actually was and if there was a way to address it. Sometimes, given the limitations we had—we were programming part of the computer's instructions directly onto a piece of hardware called a chess accelerator chip rather than writing software—there were some problems we couldn't easily fix. But often there was some way we could improve its ability to deal with a problem we had identified.

How did Deep Blue decide which moves to make?

Deep Blue was a hybrid. It had general-purpose supercomputer processors combined with these chess accelerator chips. We had software that ran on the supercomputer to carry out part of a chess computation and then hand off the more complex parts of a move to the accelerator, which would then calculate [possible moves and outcomes]. The supercomputer would take those values and eventually decide what route to take.

How did Deep Blue advance from 1996 to 1997 in order to beat Kasparov?

We did a couple of things. We more or less doubled the speed of the system by creating a new generation of hardware. And then we increased the chess knowledge of the system by adding features to the chess chip that enabled it to recognize different positions and made it more aware of chess concepts. Those chips could then search through a tree of possibilities to figure out the best move in a position. Part of the improvement between '96 and '97 is we detected more patterns in a chess position and could put values on them and therefore evaluate chess positions more accurately. The 1997 version of Deep Blue searched between 100 million and 200 million positions per second, depending on the type of position. The system could search to a depth of between six and eight pairs of moves—one white, one black—to a maximum of 20 or even more pairs in some situations. Still, *while we were confident that the 1997 Deep Blue was much better than the 1996 version, in my mind the most probable outcome of the match was a draw. Even going into the final game of the match, I was expecting a draw, and a likely rematch.*

Why didn't IBM grant Kasparov's request for a rematch after the 1997 competition?

We felt we had achieved our goal, to demonstrate that a computer could defeat the world chess champion in a match and that it was time to move on to other important research areas.

How has AI changed over the two decades since that match?

Of course, machines have improved in processing speed and memory and so on. People also started gathering—just as part of their business—a lot more data that provided fodder for the machine-learning algorithms of the day. Eventually we started realizing that combining all these things could produce some remarkable results. The IBM Watson system that played *Jeopardy!* used a machine-learning-based system that took a lot of data that existed in the world—things like Wikipedia and so on—and used that data to learn how to answer questions about the real world. Since then we have moved on to learn how to do certain kinds of perceptual tasks like speech recognition and machine vision. That has led to Watson performing more business-related tasks such as analyzing radiology images and sharing that information with physicians.

How did your experience working on Deep Blue help influence your work on AI going forward?
One thing in particular we learned is that there's more than one way to look at a complex problem. For example, in chess there's the human way, which is very pattern recognition–based and intuition-based, and then there's the machine way, which is very search intensive and looks through millions or billions of possibilities. Often these approaches are complementary. That's definitely true in chess but also in many real-world problems—that computers and humans together are better than either one alone. We wouldn't want, for example, computers to take over diagnosis and treatment of patients by themselves because there are a lot of intangibles in diagnosing a patient that are hard to capture in the data. But in terms of making recommendations about options to consider—perhaps those that are from very recent technical papers or clinical trials that maybe the doctor isn't aware of—a system like that can be very valuable.

An important part of what we're doing right now is taking very advanced artificial neural network–based systems that tend to be very black box—they aren't particularly good at explaining why they're recommending what they're recommending—and giving them the capability to explain themselves. How can you really trust a recommendation coming out of system if it can't explain it? These black box neural network systems are enormously complex, with millions of parameters in them. Part of overcoming that [complexity] may be along the lines of training a system by giving it examples of good explanations. This is particularly obvious in the health care space when a computer makes a diagnosis or recommends a treatment. If there is a reasonable explanation, then we could probably more appropriately give it the weight it deserves to help a doctor make a final decision.

ABOUT THE AUTHOR(S)

Larry Greenemeier is the associate editor of technology for *Scientific American*, covering a variety of tech-related topics, including biotech, computers, military tech, nanotech and robots.

Follow Larry Greenemeier on Twitter

Credit: Nick Higgins

Recent Articles by Larry Greenemeier

Phone Hacking Fears and Facts

IKEA-Building Robot Conquers Touchy-Feely Challenge

How Cryptojacking Can Corrupt the Internet of Things

HEALTH

Why Big Data Isn't Necessarily Better Data

Larry Greenemeier

Brainy Watson Computer to Tackle Cancer and Other Medical Research

Larry Greenemeier

Will IBM’s Watson Usher in a New Era of Cognitive Computing?

Larry Greenemeier

MIND

Integrating Left Brain and Right, on a Computer

Larry Greenemeier

Sign Up

Murray Campbell and Kasparov's blue day

Murray Campbell ('79 BSc, '81 MSc) travelled a long way from the U of A campus to take his place among the elite team of IBM scientists who designed, engineered, and programmed the first computer to defeat Garry Kasparov-the greatest chess player of all time.

Sarah Beck - 14 August 2014

Murray Campbell ('79 BSc, '81 MSc) travelled a long way from the U of A campus to take his place among the elite team of IBM scientists who designed, engineered, and programmed the first computer to defeat Garry Kasparov-the greatest chess player of all time.

The new chess champion, Deep Blue, was a looming black box designed to calculate 100 million positions per second. The computer boasted impressive processing power- and came about as the result of an immense effort on the part of the IBM team who first lost to Kasparov in 1996.

After their first defeat, Campbell and his team had 15 months to tweak Deep Blue and improve their chances, which meant enhancing the system to compute twice as many positions per second.

Campbell describes the experience of going into the last game of the six-game match. "The last game was tied-there was a lot on the line," he recalls. Kasparov was playing a risky strategy. "While Deep Blue's confidence estimates continued to climb, there was always the worry that Deep Blue was misunderstanding something." Suspended, the creators of the world's most powerful chess playing computer could only watch as Deep Blue played flawlessly to seal the victory in a brisk 19 minute game.

This iconic moment in history-man vs. machine-captured the world's attention. Reflecting on the media whirlwind, Campbell notes, "The popular documentary of*Kasparov and The Machine* was not my favorite film... it was very into the conspiracy theory." However, he praised the New York City play *The Machine* by Matthew Charman, which caught the attention of Disney for a future film about the match.

While a triumph like this may seem likely to hold its key players in suspension for all time, Campbell's scientific momentum continues to take him interesting places. Campbell continues to conduct and contribute to various projects at IBM and has an obvious passion for cognitive intelligence systems and creating smart workplaces.

Much of his current research is more interdisciplinary than some may think when imagining the frontiers of computing science. Some of these projects represent collaborations with scholars from diverse fields; for example, artificial intelligence and machine learning can be used to personalize online educational experiences in MOOCs (massive open online courses). Campbell is quick to point out, though, that each of these innovations still hinge on a strong foundation in artificial intelligence as the basis for success.

Campbell still speaks fondly of his time at the U of A, having completed both a computing science bachelor's and master's degree here. In fact, much of his early work in computer chess was conducted under the guidance of the now retired U of A professor Tony Marsland who worked for years on his own software-much of which is still available in the Department of Computing Science.

Campbell expanded on his early research while completing a PhD at Carnegie Mellon University where he created his own chess program, Deep Thought. After completing his PhD, IBM sponsored his research to continue investigating computer chess, eventually leading to the creation of Deep Blue.

With such an influential scientific career behind him, Campbell says that students today should take note that, "there's been a tremendous return in interest in artificial intelligence, and incredible progress in computers to predict and understand things that weren't possible 15 years ago. It's a really good area to get into."

Good news for U of A students who have the chance to study in the acclaimed program that has contributed influential research on chess, poker, Go, Hex, and other games using research in artificial intelligence. In fact, Campbell places the University of Alberta as "one of the strongest universities in the world for artificial intelligence using gaming as a domain."

As a member of IBM's research and development group, Campbell has played a crucial role in research for more than 20 years. In citing IBM's impact, he notes the creation of Watson, a cognitive artificial intelligence program designed to play Jeopardy. While Campbell is not personally involved in the project, he recognizes it as being on the forefront of cognitive intelligence. "It shows how the field has been developing... it's interesting to compare the game of chess with the game of Jeopardy. Chess is well defined, Jeopardy is interesting because it brings in natural language, it brings in a question and it can be a difficult question, it can have a joke in it or a pun. These questions can be arbitrarily difficult and then you have to go out and figure out what the answer is by looking at your memories," explains Campbell. "The distinction is natural language becomes fuzzy and less precise, I think that's the direction where cognitive science and cognitive computing is moving."

Note: This article originally featured in the Spring 2014 issue of Science Contours.

About the writer

Sarah Beck is a BSc Specialization student in Computing Science; she currently does undergraduate research in interactive narratives and games. In her spare time she is involved with several organizations advocating for diversity in technology and gaming. She contributes to the blog, womeningamestudies.com.

Checkers Is Solved

Jonathan Schaeffer,* Neil Burch, Yngvi Björnsson,† Akihiro Kishimoto,‡
Martin Müller, Robert Lake, Paul Lu, Steve Sutphen

The game of checkers has roughly 500 billion billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: Perfect play by both sides leads to a draw. This is the most challenging popular game to be solved to date, roughly one million times as complex as Connect Four. Artificial intelligence technology has been used to generate strong heuristic-based game-playing programs, such as Deep Blue for chess. Solving a game takes this to the next level by replacing the heuristics with perfection.

Since Claude Shannon's seminal paper on the structure of a chess-playing program in 1950 (*1*), artificial intelligence researchers have developed programs capable of challenging and defeating the strongest human players in the world. Superhuman-strength programs exist for popular games such as chess [Deep Fritz (*2*)], checkers [Chinook (*3*)], Othello [Logistello (*4*)], and Scrabble [Maven (*5*)]. However strong these programs are, they are not perfect. Perfection implies solving a game—determining the final result (game-theoretic value) when neither player makes a mistake. There are three levels of solving a game (*6*). For the lowest level, ultraweakly solved, the perfect-play result, but not a strategy for achieving that value, is known [e.g., in Hex the first player wins, but for large board sizes the winning strategy is not known (*7*)]. For weakly solved games, both the result and a strategy for achieving it from the start of the game are known [e.g., in Go Moku the first player wins and a program can demonstrate the win (*6*)]. Strongly solved games have the result computed for all possible positions that can arise in the game [e.g., Awari (*8*)].

Checkers (8 × 8 draughts) is a popular game enjoyed by millions of people worldwide, with many annual tournaments and a series of competitions that determine the world champion. There are numerous variants of the game played around the world. The game that is popular in North America and the (former) British Commonwealth has pieces (checkers) moving forward one square diagonally, kings moving forward or backward one square diagonally, and a forced-capture rule [see supporting online material (SOM) text].

Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada.

*To whom correspondence should be addressed. E-mail: jonathan@cs.ualberta.ca

†Present address: Department of Computer Science, Reykjavik University, Reykjavik, Kringlan 1, IS-103, Iceland.

‡Present address: Department of Media Architecture, Future University, Hakodate, 116-2 Kamedanakano-cho Hakodate Hokkaido, 041-8655, Japan.

The effort to solve checkers began in 1989, and the computations needed to achieve that result have been running almost continuously since then. At the peak in 1992, more than 200 processors were devoted to the problem simultaneously. The end result is one of the longest running computations completed to date.

With this paper, we announce that checkers has been weakly solved. From the starting position (Fig. 1, top), we have a computational proof that checkers is a draw. The proof consists of an explicit strategy that never loses—the program can achieve at least a draw against any opponent, playing either the black or white pieces. That checkers is a draw is not a surprise; grandmaster players have conjectured this for decades.

The checkers result pushes the boundary of artificial intelligence (AI). In the early days of AI research, the easiest path to achieving high performance was believed to be emulating the human approach. This was fraught with difficulty, especially the problems of capturing and encoding human knowledge. Human-like strategies are not necessarily the best computational strategies. Perhaps the biggest contribution of applying AI technology to developing game-playing programs was the realization that a search-intensive ("brute-force") approach could produce high-quality performance using minimal application-dependent knowledge. Over the past two decades, powerful search techniques have been developed and successfully applied to problems such as optimization, planning, and bioinformatics. The checkers proof extends this approach by developing a program that has little need for application-dependent knowledge and is almost completely reliant on search. With advanced AI algorithms and improved hardware (faster processors, larger memories, and larger disks), it has become possible to push the limits on the type and size of problems that can be solved. Even so, the checkers search space (5×10^{20}) represents a daunting challenge for today's technology.

Computer proofs in areas other than games have been done numerous times. Perhaps the best known is the four-color theorem (*9*). This deceptively simple conjecture—that given an arbitrary map with countries, you need at most four different colors to guarantee that no two adjoining countries have the same color—has been extremely difficult to prove analytically. In 1976, a computational proof was demonstrated. Despite the convincing result, some mathematicians were skeptical, distrusting proofs that had not been verified using human-derived theorems. Although important components of the checkers proof have been independently verified, there may be skeptics.

This article describes the background behind the effort to solve checkers, the methods used for achieving the result, an argument that the result is correct, and the implications of this research. The computer proof is online (*10*).

Background. The development of a strong checkers program began in the 1950s with Arthur Samuel's pioneering work in machine learning. In 1963, his program played a match against a capable player, winning a single game. This result was heralded as a triumph for the fledgling field of AI. Over time, the result was exaggerated, resulting in claims that checkers was now "solved" (*3*).

The Chinook project began in 1989 with the goal of building a program capable of challenging the world checkers champion. In 1990, Chinook earned the right to play for the World Championship. In 1992, World Champion Marion Tinsley narrowly defeated Chinook in the title match. In the 1994 rematch, Tinsley withdrew part way due to illness. He passed away eight months later. By 1996 Chinook was much

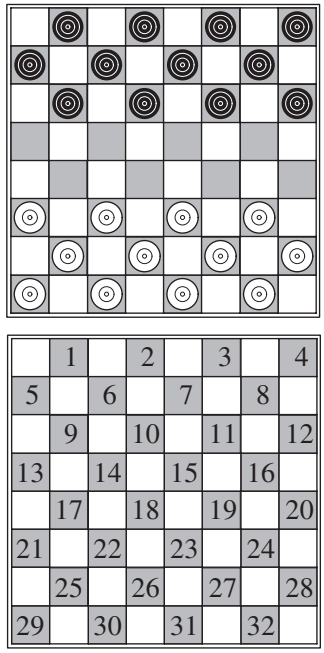

Fig. 1. Black to play and draw. (**Top**) Standard starting board. (**Bottom**) Square numbers used for move notation.

stronger than all human players, and with faster processors this gap has only grown (*3*).

Tinsley was the greatest checkers player that ever lived, compiling an incredible record that included only three losses in the period from 1950 to 1991. The unfinished Tinsley match left the question unanswered as to who was the better player. If checkers were a proven draw, then a "perfect" Chinook would never lose. As great as Tinsley was, he did occasionally make losing oversights. Hence, solving checkers would once and for all establish computers as better checkers players than all (fallible) humans.

Numerous nontrivial games have been solved, including Connect Four (*6*, *11*), Qubic (*6*), Go-Moku (*6*), Nine Men's Morris (*12*), and Awari (*8*). The perfect-play result and a strategy for achieving that result is known for these games. How difficult is it to solve a game? There are two dimensions to consider (*6*): (i) decision complexity, the difficulty of making correct move decisions, and (ii) space complexity, the size of the search space.

Checkers is considered to have high decision complexity (it requires extensive skill to make strong move choices) and moderate space complexity (5×10^{20}) (Table 1). All the games solved thus far have either low decision complexity (Qubic and Go-Moku), low space complexity (Nine Men's Morris, size 10^{11}, and Awari, size 10^{12}), or both (Connect Four, size 10^{14}).

Solving checkers. Checkers represents the most computationally challenging game solved to date. The proof procedure has three algorithm/data components (*13*): (i) Endgame databases (backward search). Computations from the end of the game back toward the starting position have resulted in a database of 3.9×10^{13} positions (all positions with ≤10 pieces on the board) for which the game-theoretic value has been computed (strongly solved). (ii) Proof-tree manager (forward search). This component maintains a tree of the proof in progress (a sequence of moves and their best responses), traverses it, and generates positions that need to be explored to further the proof's progress. (iii) Proof solver (forward search). Given a position to search by the manager, this component uses two programs to determine the value of the position. These programs approach the task in different ways, thus increasing the chances of obtaining a useful result. Figure 2 shows the forward and backward search interactions in the checkers search space.

In the manager, the proof tree can be hand-seeded with an initial line of play. From the literature (*14*), a single "best" line of play was identified and used to guide the initial foray of the manager into the depths of the search tree. Although not essential for the proof, this is an important performance enhancement. It allows the proof process to immediately focus its work on the parts of the search space that are likely to be relevant. Without it, the manager may spend unnecessary effort looking for an important line to explore. The line leads from the start of the game into the endgame databases (Fig. 2).

Backward search. Positions at the end of the game can be searched and their win/loss/draw value determined. The technique is called retrograde analysis and has been successfully used for many games. The algorithm works backward by starting at the end of the game and working toward the start. It enumerates all one-piece positions, determining their value (in this case, a trivial win for the side with the piece). Next, all two-piece positions are enumerated and analyzed. The analysis for each position eventually leads to a one-piece position with a known value, or a repeated position (draw). Next, all the three-piece positions are tackled, and so forth (SOM text). Our program has computed all the positions with ≤10 pieces on the board. The endgame databases are crucial to solving checkers. The checkers forced-capture rule quickly results in many pieces being removed from the board, giving rise to a position with ≤10 pieces—and a known value.

The databases contain the win/loss/draw result for a position, not the number of moves to a win/

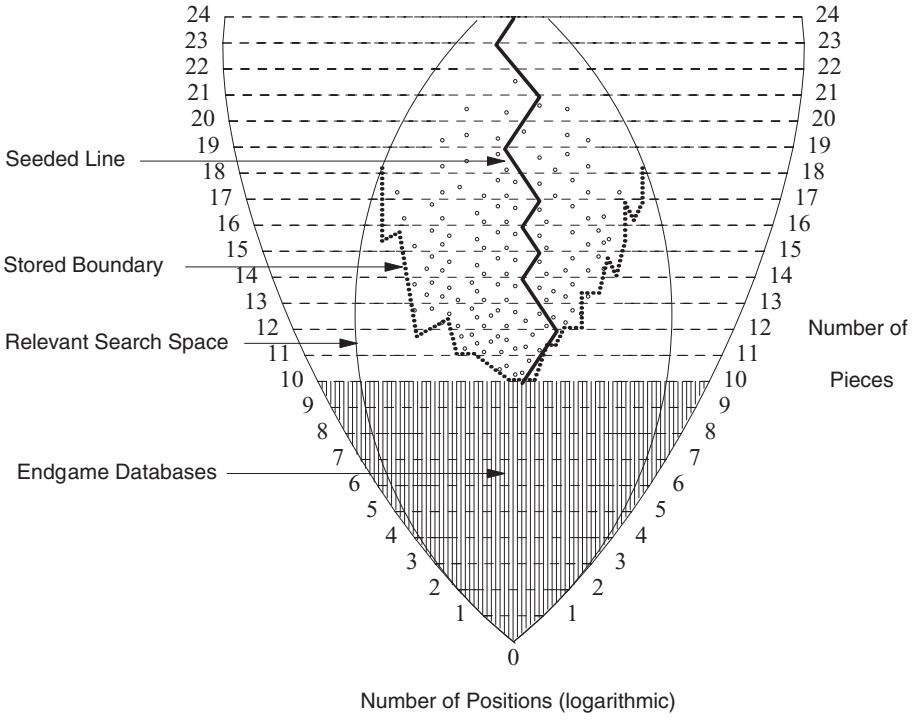

Fig. 2. Forward and backward search. The number of pieces on the board are plotted (vertically) versus the logarithm of the number of positions (Table 1). The shaded area shows the endgame database part of the proof—i.e., all positions with ≤10 pieces. The inner oval area shows that only a portion of the search space is relevant to the proof. Positions may be irrelevant because they are unreachable or are not required for the proof. The small open circles indicate positions with more than 10 pieces for which a value has been proven by a solver. The dotted line shows the boundary between the top of the proof tree that the manager sees (and stores on disk) and the parts that are computed by the solvers (and are not saved in order to reduce disk storage needs). The solid seeded line shows a "best" sequence of moves.

Table 1. The number of positions in the game of checkers. For example, the possible positions for one piece include 32 squares for the Black king, 32 squares for the White king, 28 squares for a Black checker, and 28 squares for a White checker, for a total of 120 positions.

Pieces	Number of positions
1	120
2	6,972
3	261,224
4	7,092,774
5	148,688,232
6	2,503,611,964
7	34,779,531,480
8	406,309,208,481
9	4,048,627,642,976
10	34,778,882,769,216
Total 1–10	39,271,258,813,439
11	259,669,578,902,016
12	1,695,618,078,654,976
13	9,726,900,031,328,256
14	49,134,911,067,979,776
15	218,511,510,918,189,056
16	852,888,183,557,922,816
17	2,905,162,728,973,680,640
18	8,568,043,414,939,516,928
19	21,661,954,506,100,113,408
20	46,352,957,062,510,379,008
21	82,459,728,874,435,248,128
22	118,435,747,136,817,856,512
23	129,406,908,049,181,900,800
24	90,072,726,844,888,186,880
Total 1–24	500,995,484,682,338,672,639

loss. Independent research has discovered a 10-piece database position requiring a 279-ply move sequence to demonstrate a forced win (a ply is one move by one player) (*15*). This is a conservative bound; the win length has not been computed for the more difficult (and more interesting) database positions.

The complete 10-piece databases contain 39 trillion positions (Table 1). They are compressed into 237 gigabytes, an average of 154 positions per byte. A custom compression algorithm was used that allows for rapid localized real-time decompression (*16*). This means that the backward and forward search programs can quickly extract information from the databases.

The first databases, constructed in 1989, were for less than or equal to four pieces. In 1994, Chinook used a subset of the eight-piece database for the Tinsley match (*3*). By 1996, the eight-piece database was completed, giving rise to hope that checkers could be solved. However, the problem was still too hard, and the effort came to a halt. In 2001, computer capabilities had increased substantially, and the effort was restarted. It took 7 years (1989 to 1996) to compute the original eight-piece databases; in 2001 it took only a month. In 2005, the 10-piece database computation finished. At this point, all computational resources were focused on the forward search effort.

Forward search. Development of the forward search program began in 2001, with the production version up and running in 2004. The forward search consists of two parts: the proof-tree manager, which builds the proof by identifying positions that need to be assessed, and the proof solvers, which search individual positions.

The manager maintains the master copy of the proof and uses the Proof Number search algorithm (*6*) to identify a prioritized list of positions that need to be examined. Typically, several hundred positions of interest are generated at a time so as to keep multiple computers busy. Over the past year, we used an average of 50 computers simultaneously.

The solvers get a position to evaluate from the manager. The result of a position evaluation can be proven (win, loss, or draw), partially proven (at least a draw, at most a draw), or heuristic (an estimate of how good or bad a position is). Proven positions need no further work; partially proven positions need additional work if the manager determines that a proven value is needed. If no proven information is available then the solver returns a heuristic assessment of the position. The manager uses this assessment to prioritize which positions to consider next. The manager updates the proof tree with the new information, decides which positions need further investigation, and generates new work to do. This process is repeated until a proven result for the game is determined.

The solver uses two search programs to evaluate a position. The first program (targeted at 15 s, but sometimes much longer) uses Chinook to determine a heuristic value for the position (alpha-beta search to nominal search depths of 17 to 23 ply). Occasionally, this search determines that the position is a proven win or loss. Chinook was not designed to produce a proven draw, only a heuristic draw; demonstrating proven draws in a heuristic search seriously degrades performance.

The alpha-beta search algorithm is the mainstay of game-playing programs. The algorithm does a depth-first, left-to-right traversal of the search tree (*17*) (SOM text). The algorithm propagates heuristic bounds on the value of a position: the minimum value that the side about to move can achieve and the maximum value that the side about to move can be limited to by the opponent. Lines of play that are provably outside this range are irrelevant and can be eliminated (cut off). A d-ply search with an average of b moves to consider in every position results in a tree with roughly b^d positions. In the best case, the alpha-beta algorithm only needs to examine roughly $b^{d/2}$ positions (*16*).

If Chinook does not find a proven result, then a second program is invoked (100 s). It uses the Df-pn algorithm (*18*), a space-efficient variant of Proof Number search. The search returns a proven, partially proven, or unknown result.

Algorithms based on proof numbers maintain a measure of the difficulty of proving a position. This difficulty is expressed as a proof number, a lower bound on the minimum number of positions that need to be explored to result in the position being proven. The algorithm repeatedly expands the tree below the position requiring the least effort to affect the original position (a "best-first" approach). The result of that search is propagated back up the tree, and a new best candidate to consider is determined. Proof number search was specifically invented to facilitate the proving of games. The Df-pn variant builds the search tree in a depth-first manner, requiring less computer storage.

Iterative search algorithms are commonplace in the AI literature. Most iterate on search depth (first 1 ply, then 2, then 3, etc.). The manager uses the new approach of iterating on the error in Chinook's heuristic scores (*13*). The manager uses a threshold, t, and temporarily assumes that all heuristic scores $\geq t$ are wins and all scores $\leq -t$ are losses. It then proves the result given this assumption. Once completed, t is increased to $t+\Delta$, and the process is repeated. Eventually t reaches the value of a win and the proof is complete. This iterative approach concentrates the effort on forming the outline of the proof with low values of t, and then fleshing out the details with the rest of the computation.

One complication is the graph-history interaction (GHI) problem. It is possible to reach the same position through two different sequences of moves. This means that some draws depend on the moves played leading to the duplicated position. In standard search algorithms, GHI may cause some positions to be incorrectly inferred as draws. Part of this research project was to develop an improved algorithm for addressing the GHI problem (*19*).

Correctness. Given a computation that has run for so long on many processors, an important

Table 2. Openings solved. Shown are the opening moves (using the standard square number scheme in Fig. 1, bottom), the result, the number of positions given to the solvers, and the position farthest from the start of the game that was searched (Max ply). The last two columns give the size and ply depth of the pruned minimal proof tree. Note that the total does not match the sum of the 19 openings. The combined tree has some duplicated nodes, which have been removed when reporting the total.

No.	Opening	Proof	Searches	Max ply	Minimal size	Max ply
1	09-13 22-17 13-22	Draw	736,984	56	275,097	55
2	09-13 21-17 05-09	Draw	1,987,856	154	684,403	85
3	09-13 22-18 10-15	Draw	715,280	103	265,745	58
4	09-13 23-18 05-09	Draw	671,948	119	274,376	94
5	09-13-23-19 11-16	Draw	964,193	85	358,544	71
6	09-13 24-19 11-15	Draw	554,265	53	212,217	49
7	09-13 24-20 11-15	Draw	1,058,328	59	339,562	58
8	09-14 23-18 14-23	≤Draw	2,202,533	77	573,735	75
9	10-14 23-18 14-23	≤Draw	1,296,790	58	336,175	55
10	10-15 22-18 15-22	≤Draw	543,603	60	104,882	41
11	11-15 22-18 15-22	≤Draw	919,594	67	301,310	59
12	11-16 23-19 16-23	≤Draw	1,969,641	69	565,202	64
13	12-16 24-19 09-13	Loss	205,385	44	49,593	40
14	12-16 24-19 09-14	≤Draw	61,279	45	23,396	44
15	12-16 24-19 10-14	≤Draw	21,328	31	8,917	31
16	12-16 24-19 10-15	≤Draw	31,473	35	13,465	35
17	12-16 24-19 11-15	≤Draw	23,803	34	9,730	34
18	12-16 24-19 16-20	≤Draw	283,353	49	113,210	49
19	12-16 24-19 08-12	≤Draw	266,924	49	107,109	49
Overall		Draw	Total	Max	Total	Max
			15,123,711	154	3,301,807	94

question to ask is "Are the results correct?" Early on in the computation, we realized that there were many potential sources of errors, including algorithm bugs and data transmission errors. Great care has been taken to eliminate any possibility of error by verifying all computation results and doing consistency checks. As well, some of the computations have been independently verified (SOM text).

Even if an error has crept into the calculations, it likely does not change the final result. Assume a position that is 40 ply away from the start is incorrect. The probability that this erroneous result can propagate up 40 ply and change the value for the game of checkers is vanishingly small (*20*).

Results. Our approach to solving the game was to determine the game-theoretic result by doing the least amount of work. In tournament checkers, the standard starting position (Fig. 1, top) is considered "boring," so the first three moves (ply) of a game are randomly chosen at the start. The checkers proof consisted of solving 19 three-move openings, leading to a determination of the starting position's value: a draw. Although there are roughly 300 three-move openings, more than 100 are duplicates (move transpositions). The rest can be proven to be irrelevant by an alpha-beta search.

Table 2 shows the results for the 19 openings solved to determine the perfect-play result for checkers. (Other openings have been solved but are not included here.) After an opening was proven, a postprocessing program pruned the tree to eliminate all the computations that were not part of the smallest proof tree. In hindsight, the pruned work was unnecessary, but it was not so at the time when it was assigned for evaluation. Figure 3 shows the proof tree for the first 3 ply.

The leftmost move sequence in Fig. 3 is as follows: Black moves from 09 to 13 (represented using the standard checkers notation 09-13), White replies with 22-17, and then Black moves 13-22. The resulting position has been searched and shown to be a draw (opening line 1 in Fig. 3). That means the position after 22-17 is also a draw, given that there is only one legal move available (13-22) and it is a proven draw. What is the value of the position after Black moves 09-13? To determine this, all possible moves for White have to be considered. The move 22-17 guarantees White at least a draw (at most a draw for Black). But it is possible that this position is a win for White (and a loss for Black). The remaining moves (21-17, 22-18, 23-18, 23-19, 24-19, and 24-20; opening lines 2 to 7 in Fig. 3) are all shown to be at least a draw for Black. Hence, White prefers the move 22-17 (no worse than any other move). Thus, 09-13 leads to a draw (White will move 22-17 in response).

Given that 09-13 is a draw, it remains to demonstrate that the other opening moves cannot win for Black. Note that some openings have a proven result, whereas for others only the partial result that was necessary for the proof was computed. The number of openings is small because the forced-capture rule was exploited. Opening lines 13 to 19 in Fig. 3 are needed to prove that the opening 12-16 is not a win. Actually, one opening would have sufficed (12-16, 23-19, and 16-23). However, human analysts consider this line to be a win for Black, and the preliminary analysis agreed. Hence, the seven openings beginning with the moves 12-16 and 24-19 were proven instead. This led to the least amount of computing.

There is anecdotal evidence that the proof tree is correct. Main lines of play were manually compared to human analysis (*14*), with no errors found in the computer's results (unimportant errors were found in the human analysis).

The proof tree shows the perfect lines of play needed to achieve a draw. If one side makes a losing mistake, the proof tree may not necessarily show how to win. This additional information is not necessary for proving the draw result.

The stored proof tree is only 10^7 positions. Saving the entire proof tree, from the start of the game so that every line ends in an endgame database position, would require many tens of terabytes, resources that were not available. Instead, only the top of the proof tree, the information maintained by the manager, is stored on disk. When a user queries the proof, if the end of a line of play in the proof is reached, then the solver is used to continue the line into the databases. This substantially reduces the storage needs, at the cost of recomputing (roughly 2 min per search).

The longest line analyzed was 154 ply. The position at the end of this line was analyzed by the solver, and that analysis may have gone 20 or more ply deep. At the end of this analysis is a database position, which could be the result of several hundred ply of analysis. This provides supporting evidence of the difficulty of checkers—for computers and humans.

How much computation was done in the proof? Roughly speaking, there are 10^7 positions in the stored proof tree, each representing a search of 10^7 positions (relatively small because of the extensive disk operations). Hence, 10^{14} is a good ballpark estimate of the forward search effort.

Should we be impressed with "only" 10^{14} computations? At one extreme, checkers could be solved using storage—build endgame databases for the complete search space. This would require 5×10^{20} data entries. Even an excellent compression algorithm might only reduce this to 10^{18} bytes, impractical with today's technology. This also makes it unlikely that checkers will soon be strongly solved.

An alternative would be to use only computing—i.e., build a search tree using the alpha-beta algorithm. Consider the following unreasonably optimistic assumptions: number of moves to consider is eight in noncapture positions, a game lasts 70 ply, all captures are of a single piece (23 capture moves), and the alpha-beta search does the least possible work. The assumptions result in a search tree of $8^{(70-23)} = 8^{47}$ states. The perfect alpha-beta search will halve the exponent, leading to a search of roughly $8^{47/2} \approx 10^{24}$. This would take more than a lifetime to search, given current technology.

Conclusion. What is the scientific significance of this result? The early research was devoted to developing Chinook and demonstrating superhuman play in checkers, a milestone that predated the Deep Blue success in chess. The project has been a marriage of research in AI and parallel computing, with contributions made in both of these areas. This research has been used by a bioinformatics company; real-time access of very large data sets for use in parallel search is as relevant for solving a game as it is for biological computations.

The checkers computation pushes the boundary of what can be achieved by search-intensive algorithms. It provides compelling evidence of the power of limited-knowledge approaches to artificial intelligence. Deep search implicitly uncovers knowledge. Furthermore, search algorithms are well poised to take advantage of the

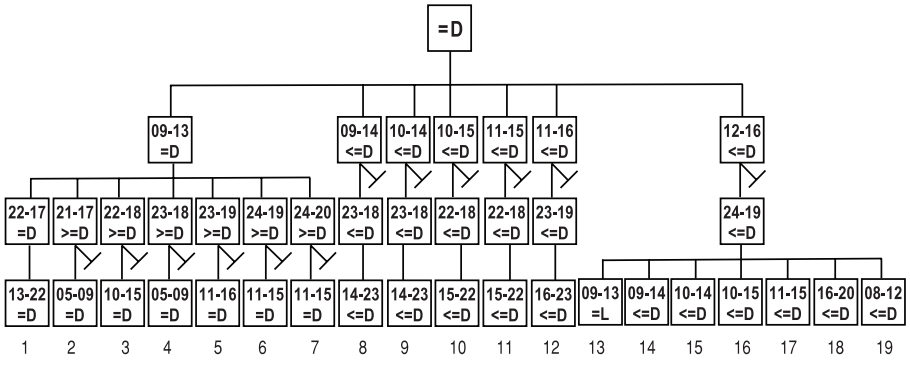

Fig. 3. The first three moves of the checkers proof tree. Move sequences are indicated using the notation from Fig. 1B, with the from-square and to-square of the move separated by a hyphen. The result of each position is given for Black, the first player to move (=D, a proven draw; =L, a proven loss; <=D, loss or draw; and >=D, draw or win). In some positions, only one move needs to be considered; the rest are cut off, as indicated by the rotated "T". Some positions have only one legal move because of the forced-capture rule.

increase in on-chip parallelism that multicore computing will soon offer. Search-intensive approaches to AI will play an increasingly important role in the evolution of the field.

With checkers finished, the obvious question is whether chess is solvable. Checkers has roughly the square root of the number of positions in chess (somewhere in the 10^{40} to 10^{50} range). Given the effort required to solve checkers, chess will remain unsolved for a long time, barring the invention of new technology. The disk-flipping game of Othello is the next popular game that is likely to be solved, but it will require considerably more resources than were needed to solve checkers (*7*).

References and Notes

1. C. Shannon, *Philos. Mag.* **41**, 256 (1950).
2. "The Duel: Man vs. Machine" (www.rag.de/microsite_chess_com).
3. J. Schaeffer, *One Jump Ahead* (Springer-Verlag, New York, 1997).
4. M. Buro, *IEEE Intell. Syst. J.* **14**, 12 (1999).
5. B. Sheppard, thesis, Universiteit Maastricht, Maastricht, Netherlands (2002).
6. V. Allis, thesis, University of Limburg, Maastricht, Netherlands (1994).
7. J. van den Herik, J. Uiterwijk, J. van Rijswijck, *Artif. Intell.* **134**, 277 (2002).
8. J. Romein, H. Bal, *IEEE Computer* **36**, 26 (2003).
9. K. Appel, W. Haken, *Sci. Am.* **237**, 108 (1977).
10. Chinook Web site, with proof (www.cs.ualberta.ca/~chinook).
11. John's Connect Four Playground (http://homepages.cwi.nl/~tromp/c4/c4.html).
12. R. Gasser, thesis, ETH Zürich, Switzerland (1995).
13. J. Schaeffer *et al.*, "Solving Checkers" (www.ijcai.org/papers/0515.pdf).
14. R. Fortman, *Basic Checkers* (http://home.clara.net/davey/basicche.html).
15. "Longest 10PC MTC" (http://pages.prodigy.net/eyg/Checkers/longest-10pc-mtc.htm).
16. J. Schaeffer *et al.*, in *Advances in Computer Games*, J. van den Herik, H. Iida, E. Heinz, Eds. (Kluwer, Dordrecht, Netherlands, 2003), pp. 193–210.
17. D. Knuth, R. Moore, *Artif. Intell.* **6**, 293 (1975).
18. A. Nagai, thesis, University of Tokyo, Japan (2002).
19. A. Kishimoto, M. Müller, in *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (AAAI Press, Menlo Park, CA, 2004) pp. 644–649.
20. D. Beal, thesis, Universiteit Maastricht, Maastricht, Netherlands (1999).

21. The support of Canada's Natural Sciences and Engineering Research Council (NSERC), Alberta's Informatics Circle of Research Excellence (iCORE), and the Canada Foundation for Innovation is greatly appreciated. Numerous people contributed to this work, including M. Bryant, J. Culberson, B. Gorda, B. Knight, D. Szafron, K. Thompson, and N. Treloar.

Supporting Online Material

www.sciencemag.org/cgi/content/full/1144079/DC1
Materials and Methods
Figs. S1 to S4
References

20 April 2007; accepted 6 July 2007
Published online 19 July 2007;
10.1126/science.1144079
Include this information when citing this paper.

TLR3 Deficiency in Patients with Herpes Simplex Encephalitis

Shen-Ying Zhang,[1,2,3] Emmanuelle Jouanguy,[1,2,3] Sophie Ugolini,[4] Asma Smahi,[5] Gaëlle Elain,[6] Pedro Romero,[7] David Segal,[8] Vanessa Sancho-Shimizu,[1,2] Lazaro Lorenzo,[1,2] Anne Puel,[1,2] Capucine Picard,[1,2,9] Ariane Chapgier,[1,2] Sabine Plancoulaine,[1,2] Matthias Titeux,[10] Céline Cognet,[4] Horst von Bernuth,[1,2] Cheng-Lung Ku,[1,2] Armanda Casrouge,[1,2] Xin-Xin Zhang,[3] Luis Barreiro,[11] Joshua Leonard,[8] Claire Hamilton,[1,2] Pierre Lebon,[12] Bénédicte Héron,[13] Louis Vallée,[14] Lluis Quintana-Murci,[11] Alain Hovnanian,[10] Flore Rozenberg,[12] Eric Vivier,[4] Frédéric Geissmann,[6] Marc Tardieu,[15] Laurent Abel,[1,2] Jean-Laurent Casanova[1,2,3,16]*

Some Toll and Toll-like receptors (TLRs) provide immunity to experimental infections in animal models, but their contribution to host defense in natural ecosystems is unknown. We report a dominant-negative *TLR3* allele in otherwise healthy children with herpes simplex virus 1 (HSV-1) encephalitis. TLR3 is expressed in the central nervous system (CNS), where it is required to control HSV-1, which spreads from the epithelium to the CNS via cranial nerves. TLR3 is also expressed in epithelial and dendritic cells, which apparently use TLR3-independent pathways to prevent further dissemination of HSV-1 and to provide resistance to other pathogens in TLR3-deficient patients. Human TLR3 appears to be redundant in host defense to most microbes but is vital for natural immunity to HSV-1 in the CNS, which suggests that neurotropic viruses have contributed to the evolutionary maintenance of TLR3.

The contribution of Toll and Toll-like receptors to immunity has been studied extensively in the past decade. Toll-deficient *Drosophila* were shown to be susceptible to experimental infections with certain fungi in 1996 (*1*), and a Toll-like receptor 4 (TLR4) null mutation in mice resistant to lipopolysaccharide (LPS) but susceptible to certain Gram-negative bacteria was identified in 1998 (*2*). Mice deficient for individual TLRs have since been generated and shown to have diverse infectious phenotypes, from susceptibility to resistance, depending on the TLR-pathogen combination (*3*). However, it remains unclear whether TLRs play nonredundant roles—beneficial or detrimental—in natural, as opposed to experimental, infections. This biological question is important, because natural selection acts on a given species in the setting of natural (rather than experimental) ecosystems. The human model is particularly suitable for analyses of the relevance of genes such as those of TLRs to host defense in natural ecosystems (*4*). Nevertheless, although many studies have suggested that TLR genes are involved in human infectious diseases, this has not been unambiguously demonstrated (*5*). In particular, no primary immunodeficiency involving TLRs has been identified.

The discovery of inherited interleukin 1 receptor-associated kinase-4 (IRAK-4) deficiency in children with bacterial diseases implicated human TLRs, interleukin-1 receptors (IL-1Rs), or both in host defense (*6*, *7*). However, the narrow range of infections documented in such patients indicates that IRAK-4–dependent, TLR-mediated immunity is redundant for protective immunity to most microbes. In particular, IRAK-4–deficient patients are not susceptible to herpes simplex virus 1 (HSV-1) encephalitis (HSE). In HSE, HSV-1 infects epithelial cells in the oral and nasal mucosa and progresses to the central nervous system (CNS) via the trigeminal or olfactory nerves (*8*). A genetic etiology of HSE was found in two children who lacked functional UNC-93B (*9*), an endoplasmic reticulum protein required for TLR3, TLR7, TLR8, and TLR9 signaling (*10*). Both UNC-93B– and IRAK-4–deficient patients fail to signal through TLR7, TLR8, and TLR9, but unlike IRAK-4–deficient patients (*7*), UNC-93B–deficient patients display impaired TLR3-dependent interferon-α (IFN-α) -β, and -λ production (*9*). Moreover, HSV-1 is a double-stranded DNA virus with double-stranded RNA (dsRNA) intermediates (*11*), and TLR3 recog-

[1]Human Genetics of Infectious Diseases, Institut National de la Santé et de la Recherche Médicale (INSERM), U550, Faculty Necker, Paris 75015, France. [2]University Paris René Descartes, Paris 75015, France. [3]French-Chinese Laboratory of Genetics and Life Sciences, Rui Jin Hospital, Shanghai Jiao Tong University, Shanghai 200025, China. [4]Marseille-Luminy Immunology Institute, Marseille 13288, France. [5]Department of Genetics, INSERM, U781, Necker Hospital, Paris 75015, France. [6]Laboratory of Mononuclear Cell Biology, INSERM, U838, Necker Hospital, Paris 75015, France. [7]Ludwig Institute for Cancer Research, Lausanne Branch, University Hospital, Lausanne 1005, Switzerland. [8]Experimental Immunology Branch, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA. [9]Center for the Study of Immunodeficiencies, Necker Hospital, Paris 75015, France. [10]INSERM, U563, University Toulouse Paul Sabatier, Toulouse 31000, France. [11]Centre National de la Recherche Scientifique, URA3012, Pasteur Institute, Paris 75015, France. [12]Virology, Cochin-Saint-Vincent de Paul Hospital, University Paris René Descartes, Paris 75014, France. [13]Pediatric Neurology, Trousseau Hospital, Paris 75012, France. [14]Pediatric Neurology, University Hospital, Lille 59037, France. [15]Pediatric Neurology, Bicêtre Hospital, University Paris Sud, Kremlin-Bicêtre 94270, France. [16]Pediatric Hematology-Immunology, Necker Hospital, Paris 75015, France.

*To whom correspondence should be addressed. E-mail: casanova@necker.fr

Checkers Is Solved

Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu and Steve Sutphen

(July 19, 2007)
Science **317** (5844), 1518-1522. [doi: 10.1126/science.1144079]
originally published online July 19, 2007

Editor's Summary

Science (print ISSN 0036-8075; online ISSN 1095-9203) is published weekly, except the last week in December, by the American Association for the Advancement of Science, 1200 New York Avenue NW, Washington, DC 20005.