# Algorithmic Progress in Six Domains

Katja Grace
*MIRI Visiting Fellow*

## Abstract

We examine evidence of progress in six areas of algorithms research, with an eye to understanding likely algorithmic trajectories after the advent of artificial general intelligence. Many of these areas appear to experience fast improvement, though the data are often noisy. For tasks in these areas, gains from algorithmic progress have been roughly fifty to one hundred percent as large as those from hardware progress. Improvements tend to be incremental, forming a relatively smooth curve on the scale of years.

Grace, Katja. 2013. *Algorithmic Progress in Six Domains*. Technical report 2013-3. Berkeley, CA: Machine Intelligence Research Institute. Last modified December 9, 2013.

## Contents

## List of Figures

## List of Tables

## 1. Introduction

In the future, it is anticipated that the smartest creatures will have software minds. At that point, their intelligence may improve quickly—as quickly as the underlying algorithms can be improved (Muehlhauser and Salamon 2012; Yudkowsky 2013).

To understand how progress will proceed when intelligence becomes an algorithmic problem, we might do well to look at how existing algorithms improve.

Several features of characteristic algorithmic improvement curves should interest us. How fast is progress? Does it tend to be smooth or lumpy? Is there rapid progress followed by diminishing returns, or does progress accelerate? How much of progress comes immediately from large insights, and how much comes from tinkering later on? Are there general patterns to be found?

Progress on human-level AI may be faster than progress on historical algorithmic problems, because superhuman intelligences are likely to be more productive researchers than their human counterparts. Moreover, mature AI will have unprecedented economic value, and hence probably attract commensurately more investment. The population will be larger, and technology will be better. As a result, raw improvement rates are not necessarily strong indicators of what will happen after human-level AI is developed. There is, however, great variation in how much effort is invested in different contemporary problems, so investigating the current relationships between inputs and outputs in algorithms should give us a better picture of what to expect when inputs are changed in the future.

If we know more about the work that produced the algorithmic improvements, we may be able to answer some other useful questions. How does the rate of progress depend on the number of people working in an area? How much does information sharing matter?

All these questions contribute to what we might expect in an intelligence explosion. Speed tells us about the timescale over which we should expect an intelligence explosion. If we see that algorithms robustly become twice as good in half a year to five years, we might expect an intelligence explosion to take half a year to five years. Whether progress speeds up or slows down, and how fast such changes occur, might help us predict whether a sustained feedback loop could appear, whether parties who get ahead will tend to stay ahead, and consequently whether such a transition is likely to be destructive. The sizes of jumps in capability and the importance of large insights also say a lot about how abrupt a transition might be, and the extent to which small competing entities can get ahead of the competition and stay forever ahead. The extent to which progress tends to be predictable tells us something about how likely an intelligence explosion is to be surprising. If insights translate quickly to improvements in capabilities, this will allow lucky entities to get ahead more than if gains from insights are returned spread out over the long run. The effects of sharing between groups tell us something about whether information transfer should undermine local takeoff scenarios. The extent to which scaling up the size of a research team scales up progress tells us about the potential for feedback from the creation of AI to AI research.

Each of these questions deserves a thoroughly researched answer, but much of the information is probably visible in the first glance. This paper is a collection of first glances: roughly what the history of algorithmic improvement looks like, focusing on areas where improvement can be easily measured. Looking at what is readily apparent raises some issues with selection effects, which we shall visit in the next section. This paper will neither analyze the data extensively nor attempt to point out all of its interesting implications.

## 2. Summary

This document presents empirical evidence regarding progress in six areas of algorithms research. This section summarizes these findings, while the rest of the document details them. Skimming the figures in this document should also constitute a good summary.

In recent *Boolean satisfiability* (SAT) competitions, SAT solver performance has increased 5–15% per year, depending on the type of problem. However, these gains have been driven by widely varying improvements on particular problems. Retrospective surveys of SAT performance (on problems chosen after the fact) display significantly faster progress.

*Chess programs* have improved by around fifty Elo points per year over the last four decades. Estimates for the significance of hardware improvements are very noisy but are consistent with hardware improvements being responsible for approximately half of all progress. Progress has been smooth on the scale of years since the 1960s, except for the past five.

*Go programs* have improved about one stone per year for the last three decades. Hardware doublings produce diminishing Elo gains on a scale consistent with accounting for around half of all progress.

Improvements in a variety of *physics simulations* (selected after the fact to exhibit performance increases due to software) appear to be roughly half due to hardware progress.

The *largest number factored* to date has grown by about 5.5 digits per year for the last two decades; computing power increased ten-thousand-fold over this period, and it is unclear how much of the increase is due to hardware progress.

Some *mixed integer programming* (MIP) algorithms, run on modern MIP instances with modern hardware, have roughly doubled in speed each year. MIP is an important optimization problem, but one which has been called to attention after the fact due to performance improvements. Other optimization problems have had more inconsistent (and harder to determine) improvements.

Various forms of *machine learning* have had steeply diminishing progress in percentage accuracy over recent decades. Some vision tasks have recently seen faster progress.

## 3. A Few General Points

### 3.1. On Measures of Progress

This report deals mostly with progress on well-defined, relatively easy to measure axes, such as speed and accuracy. Such progress is not the only goal of algorithm developers; algorithms are also optimized to be general, simple to maintain, and usable, among other desiderata. These things will sometimes be traded off against one another, so algorithms are not as fast as they would be if speed were the only goal, though it is not obvious how this affects relative improvements. Some of the data here is from competitions, which presumably helps to direct efforts toward specific, measurable benchmarks. Even so, developers have goals beyond winning the present competition.

It is often challenging to determine whether progress is speeding up or slowing down. Progress that is slowing by one measure is always accelerating by some other measure, so such claims are highly dependent on context and on the performance benchmarks we care about. For instance, if you double the speed of a chess algorithm, the number of levels searched grows logarithmically, the number of board positions examined grows exponentially, and the Elo rating grows linearly. It is tempting to use the most intuitive measure, such as percentage instead of logarithm of percentage, but this is not always related to what we really care about. For instance, progress in machine translation appears to have slowed in terms of percentage accuracy, but improving accuracy by a percentage point in the 90% regime arguably increases the usefulness of translation software much more than improving accuracy by a point in the 30% regime. If we are trying to predict whether a process of self-improvement would accelerate exponentially, the interesting measure is probably the same one with which we measure inputs.

### 3.2. Inputs and Outputs

Ideally, we would like to know the relationships between inputs and outputs for algorithmic progress. For instance, we would like to know how much improvement is produced by one year of an intelligent person's time. Inputs are generally harder to measure than outputs; even if you know that a chess project belongs to one person, it isn't clear how much of their time they spend on it, and how much of the relevant added insight they produce personally rather than harvesting from a larger community. Consequently, most of the data collected here is about outputs.

Outputs alone can still tell us a lot, especially as we already know some basic things about inputs. For instance, even if we are unsure how many people work on a problem, or how much of the intellectual work is done outside the apparent group working on it, we can probably assume that these remain roughly stable over short spans of years. So, if we see jumpy progress, we can tentatively attribute it to something else. Even if we are unsure which hardware is used for particular applications, we know the rough rate at which hardware gets better, and we can often assume state-of-the-art hardware is being used. To estimate progress in software, we can take overall progress and subtract the approximate contributions from hardware.

## 3.3. On Selection

The most salient algorithmic problems might be those for which progress is particularly fast (or slow), so looking at algorithms that are being reported on might give us a biased impression of the overall rate of progress. There are a few ways to mitigate this difficulty. In general, we should pay attention to motives of authors. Reports such as "A Science-Based Case for Large-Scale Simulation" (Keyes et al. 2004) might intend to illustrate points by highlighting fast-improving algorithms. The existence of benchmarks introduces a more ambiguous selection effect, which could nevertheless significantly bias results. In many cases, we should treat estimates as being optimistic rather than representative. We should rely more on assessments that are planned in advance of knowledge about performance. Competitions are better than retrospective analyses, and problems that were singled out early are better than problems that were selected after some progress. For many purposes, it doesn't matter much whether the sample is representative; trends that hold robustly over a range probably continue to hold outside of that range, and an optimistic sample can give us a better picture of the high end of progress, as long as we interpret it correctly.

That which is easily measured may improve faster than more nebulous qualities, particularly if such measures are being used to guide progress. Yet we may care about the nebulous qualities. For instance, it is easier to measure progress in voice recognition in specific circumstances than it is to measure the breadth of circumstances in which it works well, weighted by how likely we are to want to use it there. Thus, progress on well-defined metrics, such as most of what we will examine here, will tend to overestimate the progress we care about.

## 4. Boolean Satisfiability (SAT)

### 4.1. SAT Solving Competition

The Boolean satisfiability (SAT) solving competition has been held once every year or two since 2002 and publishes detailed results. For instance, the 2007 competition produced 78,507 data points of time taken by a particular solver on a particular SAT instance (problem) on fixed hardware. Some instances are taken from industrial applications, some are handcrafted, and some are produced randomly from a distribution known to be on the boundary of what is possible for existing algorithms. Each problem instance is solved by a number of competing programs. This section presents data from the 2007, 2009, and 2011 competitions.

Over the three competitions investigated here, there were 190,074 data points for 3,015 different instances. The instances used each year change substantially;[1] however, there is overlap. Many specific industrial and handcrafted instances return in multiple contests. Problems from the same random distributions also appear in multiple contests, though never exactly the same problems.

Problems from the competitions are published, so it is possible that some progress when problems are repeated comes from algorithms being optimized for known problems. This cannot be a factor in progress on the random instances, as they are changed every year. Progress was greater for the random instances than for the handcrafted and industrial instances, which suggests this is not a big effect.

The times recorded are total processing times (if multiple processors are used, the times spent by each are added together). The same hardware was used between 2007 and 2009: Intel Xeon 3.00 GHz processors with 2 GB of RAM and a 2 MB cache. In 2011, the hardware was changed to Intel Xeon 2.67 GHz cores with 32 GB of RAM and an 8 MB cache.[2] The later processor is newer and has larger cache and RAM, so it may be somewhat faster overall, notwithstanding the slower clock speed.

Every competition has time and memory limits. These are not necessarily the same between competitions, or between sections of the same competition. Whether a solver timed out or "memoried out" on a problem is recorded. In the following, timeouts are treated as if they had taken infinite time to solve, but are excluded when calculating average rates of progress (except via their effect on median times). This ensures that changes in the set of feasible problems don't directly bias the results.

1. One might fear selection bias here—e.g., problems may be disproportionately kept around for another year if people anticipate they will remain in a particular band of difficulty.

2. For more extensive hardware details, see "launcher data" on benchmark pages, for instance in 2009 or 2011.

### 4.1.1. Industrial and Handcrafted SAT Instances

Of 1,294 industrial and handcrafted problems in the three-competition (five-year) data-set, 279 were used in more than one competition, and fourteen were used in all three competitions. We will focus on the 189 that were used in two consecutive competitions and that were solved by at least one competitor.

#### Speedup Distribution

Figure 1 shows how much the best time improved on each problem between consecutive contests (that is, over two years).[3] As we can see, the distribution is almost uniform between zero and one, with a small fraction taking much longer than before. There is a flat spot: around six percent of problems' times changed by less than one percent between years.

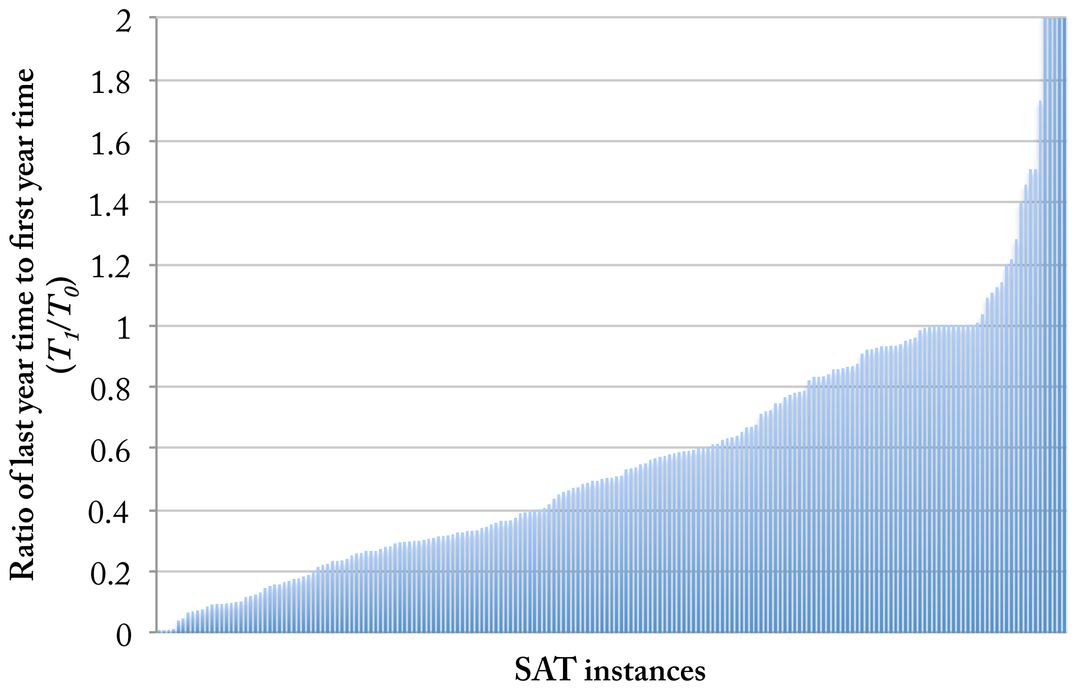


Figure 1: Reduction in time taken for handcrafted and industrial Boolean satisfiability (SAT) instances over two years. The horizontal axis represents each of the 189 SAT instances compared, ordered from largest to smallest improvement. The vertical axis represents the ratio of time to solve in later competition ($T_1$) to time to solve in earlier competition ($T_0$). The graph is truncated at 2, though a few problems took two to ten times longer in the second year.

For reference, figures 2–4 show the distribution of times in the first year of each of the periods to be considered (2007–2009, 2009–2011, and these combined). Problems

3. Figures 1–10 and Tables 1–4 were created using data from the SAT Competition Organizers (2007–2011).

do not appear in the first-year times if they were not solved by any solver in the second year.

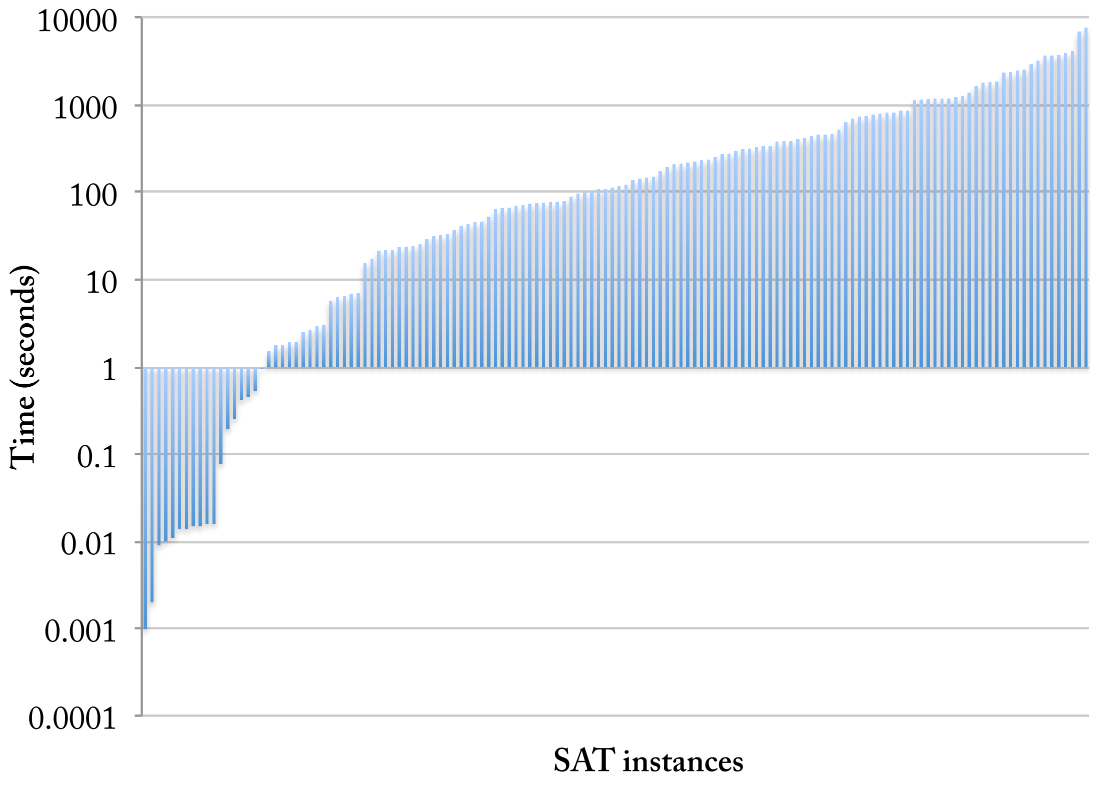


Figure 2: Fastest 2007 times to solve handcrafted and industrial SAT instances completed in both 2007 and 2009, ordered from fastest to slowest.

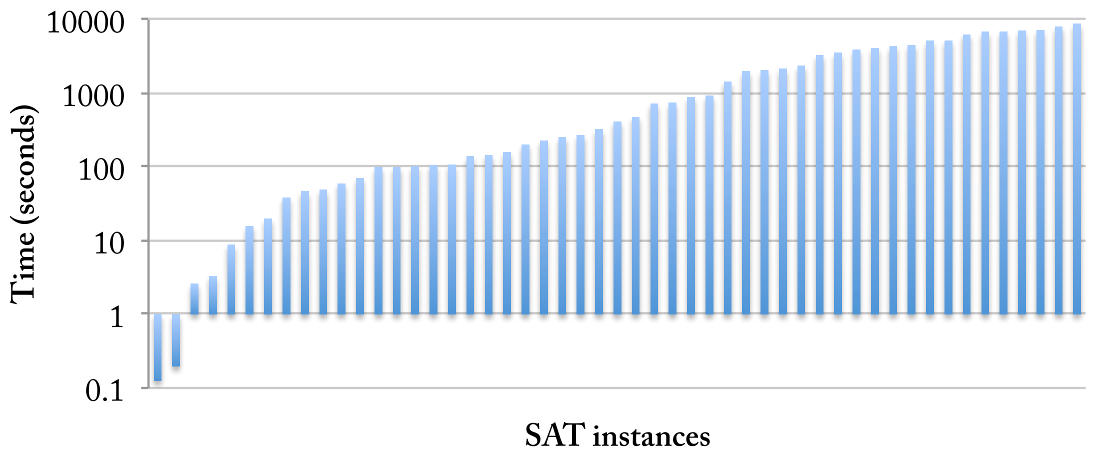


Figure 3: Fastest 2009 times to solve handcrafted and industrial SAT instances completed in both 2009 and 2011, ordered from fastest to slowest.

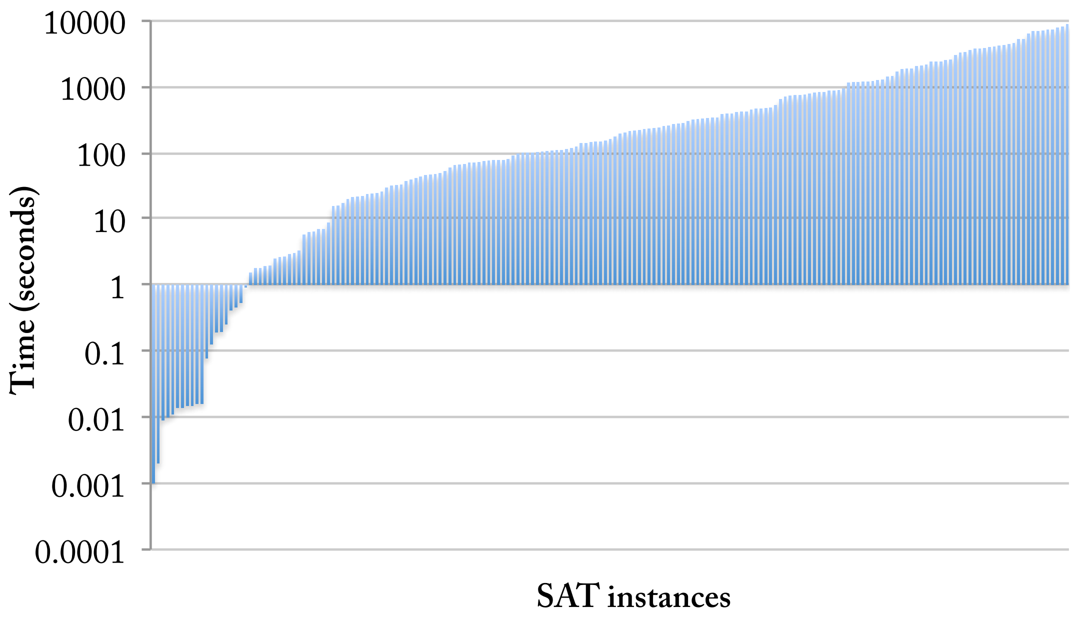


Figure 4: Fastest first-year times to solve handcrafted and industrial SAT instances completed in two consecutive competitions between 2007 and 2011, ordered from fastest to slowest.

## Two-Year Improvements

In order to measure overall improvement in a period, we need to combine the speedup on many separate problems into a single indicator. This aggregation requires making some assumptions about the distribution of problems that we care about. As can be seen in figure 4, the competition uses problems with a broad distribution of difficulties, spread across about six orders of magnitude. This means the total time to solve all of the benchmarks will mostly depend on how much the most time-consuming benchmarks got better, which is not a realistic indicator of the progress we are interested in. This approximately uniform distribution over many orders of magnitude suggests the competition organizers consider problems of different difficulties to be similarly important. If we want to give equal weight to each problem in measuring the speedup, a better way is to use a bundle of problems where we initially spend an equal amount of time on every problem. The speedup on this bundle is just the mean time taken in the second period when the first-period time is normalized to one unit. Table 1 shows this number for 2007–2009, 2009–2011, and both of these combined.

This scheme introduces a subtle bias into our estimates. Ideally, different SAT instances would be weighted by underlying characteristics that determine their difficulty. But, especially in the case of random problems, the actual time required to solve a problem is a noisy indicator of its difficulty. The proposed weighting system tends to overweight problems that happened to be solved unusually quickly, incorrectly supposing that such problems appear more often in a representative basket. This means that this

Table 1: Overall improvements between consecutive competitions, in terms of mean fraction of time taken to solve a problem in a later year ($T_1$) over the time taken in an earlier year ($T_0$).

| | 2007–2009 | 2009–2011 | All |
|---|---|---|---|
| Mean $T_1/T_0$ | 0.75 | 0.41 | 0.65 |
| Annual reduction | 13% | 36% | 19% |

weighting system systematically advantages earlier years, by overweighting problems that happened to be solved quickly in the first year. As a consequence, the stated rate of progress will be systematic underestimates of the real rate of progress. This effect will be most pronounced when there is independent random variation in the solution times for different instances.

As we can see, problems took on average 35% less time to solve after two years, or 19% less per year. Annual improvement in the 2009–2011 period was almost three times that in the 2007–2009 period.[4]

## Difficulty and Progress

Here we will look at the relationship between how long a problem initially takes and how much this time is reduced in the following two years. This tells us whether progress is overwhelmingly driven by very difficult things becoming easy, or by smaller improvements across the range of problem difficulties. These would be quite different situations, and might provide useful information about other important questions—for example, the possibility of a sustained intelligence explosion or abrupt changes in capabilities.

The relationship between difficulty and improvement should also give us a better picture of whether progress on specific problems gets faster or slower, without needing a long series of data. If problems that are already quicker have reliably more improvement, then we should expect to see accelerating progress on any given problem. On the other hand, if it is the harder problems that improve more, we should see declining progress on each problem.

The initial solution times fall roughly within six orders of magnitude (see figure 4). We can divide them into three buckets of roughly two orders of magnitude each. Table 2 shows time improvement for each of these buckets, in each period, using the same measure as above. Figure 5 shows the entire distribution of initial speed and

4. Note that the processor speed was slightly reduced in 2011, although available RAM and cache were increased and the processor was newer. The difference between these annual speedups is probably significantly greater than can be explained by the change in hardware between 2009 and 2011.

Table 2: Improvements by problem difficulty. Buckets contain problems of different lengths ($T_0$ denotes the earlier time). For each bucket and time period the table shows (1) the mean fraction of time taken in the second competition compared to the first competition, (2) the number of problems in that class, and (3) the annual time reduction for that bucket and period.

| | 2007–2009 | | | 2009–2011 | | | All | | |
|---|---|---|---|---|---|---|---|---|---|
| Bucket 1 ($T_0 < 1$s) | 0.85 | 18 | 8% | 0.63 | 2 | 21% | 0.83 | 20 | 9% |
| Bucket 2 (1s $< T_0 <$ 100s) | 0.77 | 47 | 12% | 0.63 | 12 | 21% | 0.74 | 59 | 14% |
| Bucket 3 (100s $< T_0$) | 0.71 | 73 | 16% | 0.32 | 37 | 43% | 0.58 | 110 | 24% |

improvement. Both suggest that problems that initially take longer tend to see faster improvement. Though this effect is large, it is not overwhelmingly so—the easiest problems in each competition see approximately half as much annual improvement as the most difficult ones.

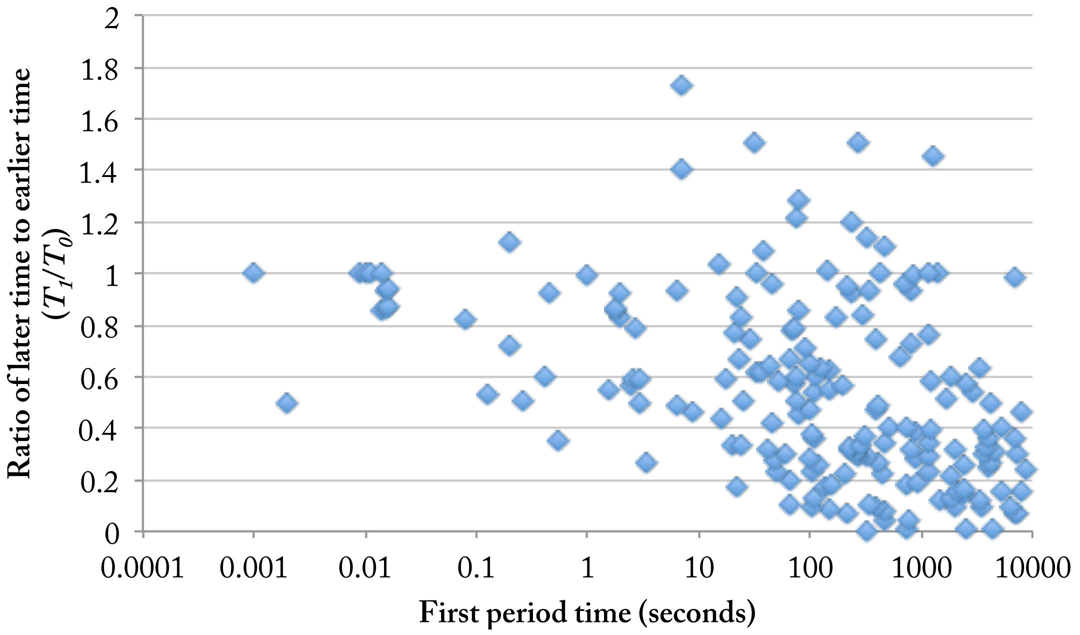


Figure 5: Initial difficulty and fractional improvement over two years for each handcrafted or industrial SAT problem. The horizontal axis shows how long the problem initially took; the vertical axis shows the ratio of its time in the second competition ($T_1$) to its time in the first competition ($T_0$). We exclude five outliers that initially took a long time and got up to ten times worse.

One might try to explain the different speedups in 2007–2009 and 2009–2011 by changes in the distribution of problems. Compared to the first period, the second period has disproportionately slower problems, which tend to see more improvement. This reasoning doesn't straightforwardly work, as each individual bucket in table 2 also sees more improvement in the second period. However, the distribution of problems inside buckets also changes between periods, seemingly also toward longer problems in the second period.

### 4.1.2. Random SAT Instances

The same distributions of random problems were included in consecutive competitions, though new random problems were used each year. For each distribution, there are a number of specific problems from that distribution each year. For each individual problem, we find the fastest algorithm for that problem. Because the same problems are not included in consecutive competitions, it is not trivial to make comparisons between years. One approach would be to compare the *best time* for the *median problem*. In fact, this works very poorly, because the distributions are chosen such that roughly half of the problems are satisfiable and half are unsatisfiable, and these two categories require very different amounts of time. This means that the median will jump around a lot depending on whether it falls just in the satisfiable half or just in the unsatisfiable half. To get roughly the same information while avoiding this problem, we will look at 25th percentile and 75th percentile times; that is, for each problem type, in each year, we will find the minimum times to solve every problem of that type, and take the 25th percentile time. Nevertheless, different problems from the same distribution will still have different difficulties, and this will complicate year-to-year comparisons.

#### Overall Picture

There were only twenty-five random problem types solved in more than one contest. A few more were used, but could not be finished in more than one contest. Figures 6 and 7 show the best times for the 25th percentile problem of each type and the 75th percentile problem of each type, respectively.

#### Speedups for Individual Problem Types

Figures 8 and 9 show speedups in order of size for all problems solved in consecutive competitions (some problems will be repeated, because they were solved in all years).

#### Two-Year Improvements

Table 3 shows mean improvements in each two-year interval. As with the handcrafted and industrial problems, there was much more improvement at both percentiles in the second interval. Again, the harder problems saw bigger fractional improvements in time.

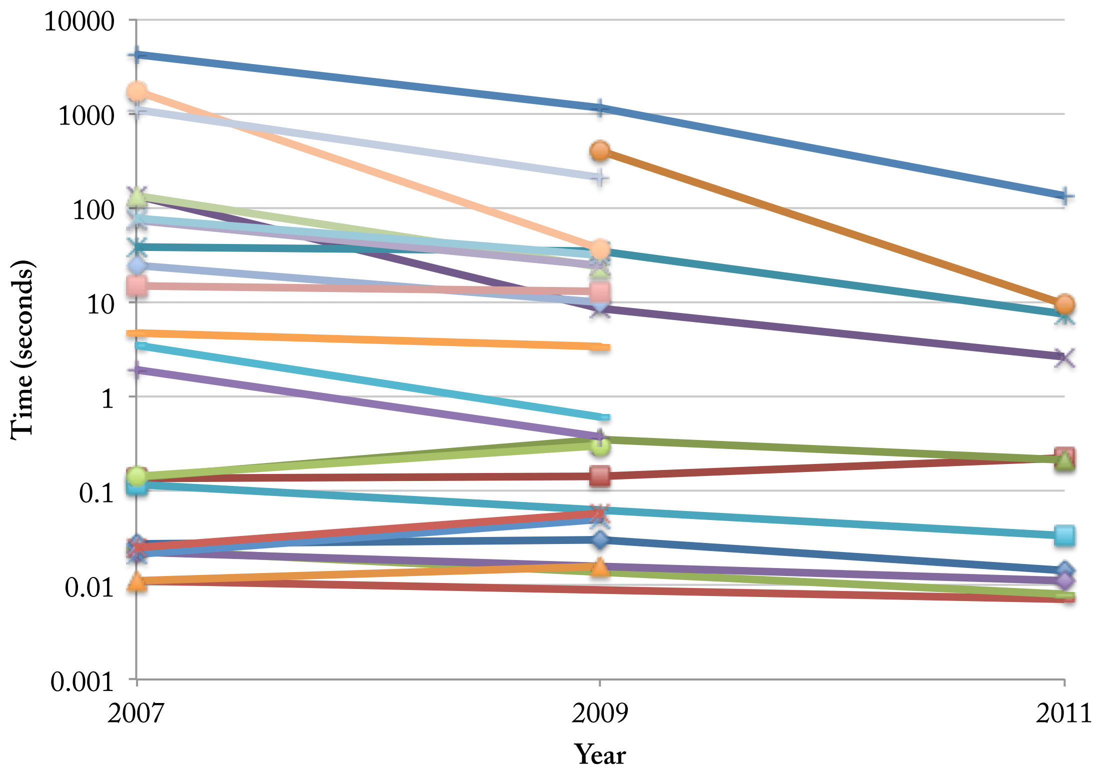


Figure 6: The 25th percentile of best solve times for all types of random SAT problems solved in more than one contest. Each line represents a different problem type.

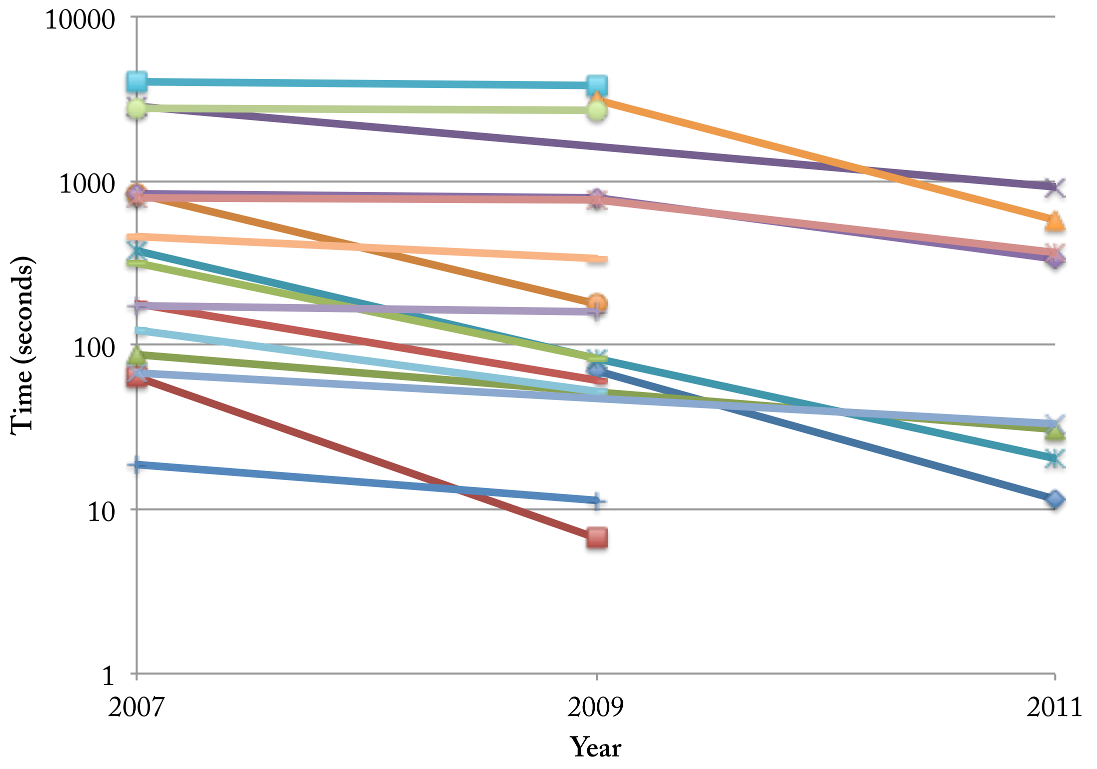


Figure 7: The 75th percentile of best solve times for all types of random SAT problems solved in more than one contest. Each line represents a different problem type.

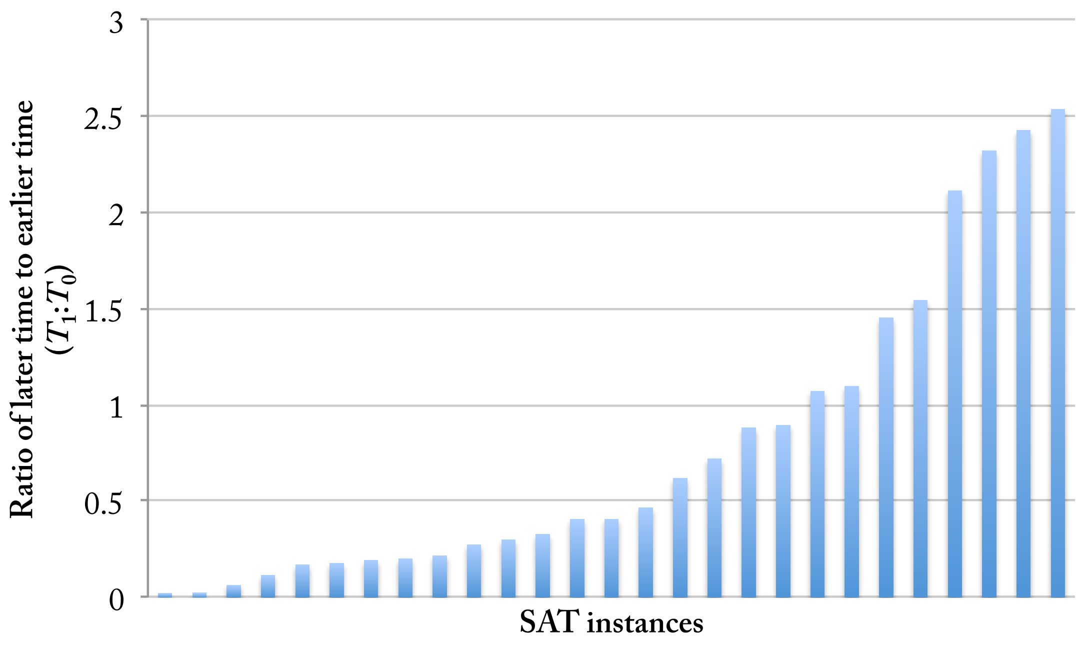


Figure 8: Fractional change in 25th percentile best times over two years, ordered from largest to smallest improvement. Note that a number greater than one indicates slowdown.

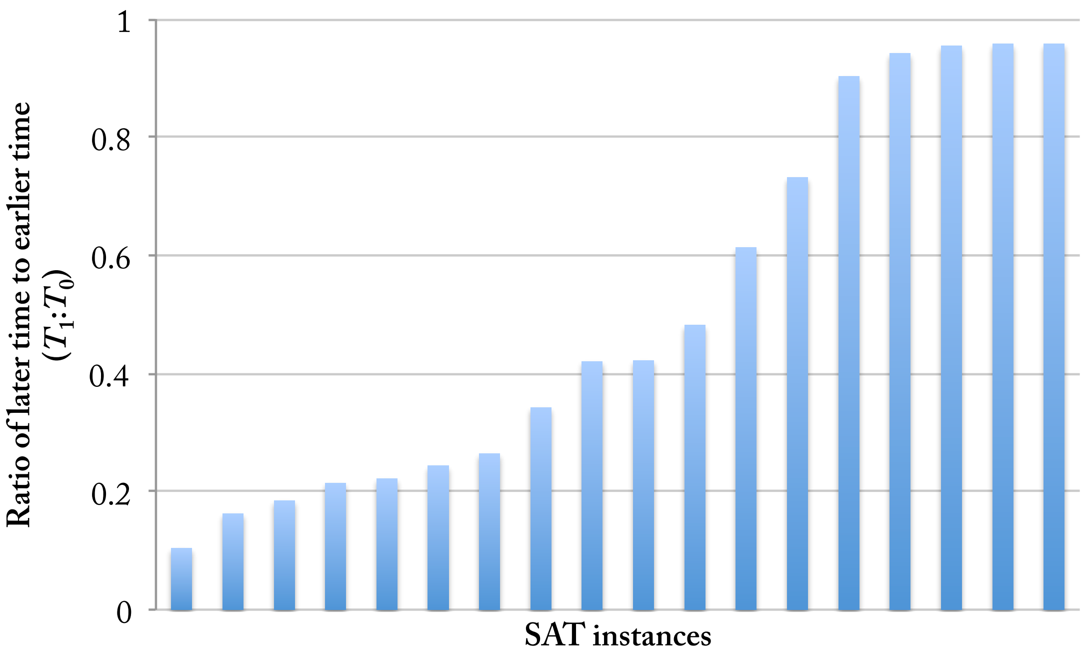


Figure 9: Fractional change in 75th percentile best times over two years, ordered from largest to smallest improvement.

Table 3: Mean improvements by problem type difficulty using the 25th and 75th percentile of best times. Buckets contain problems of different lengths ($T_0$ denotes the earlier time).

| Period | 2007–2009 | | 2009–2011 | | All | |
|---|---|---|---|---|---|---|
| Percentile | 25th | 75th | 25th | 75th | 25th | 75th |
| Mean $T_1/T_0$ | 0.89 | 0.59 | 0.47 | 0.30 | 0.78 | 0.51 |
| Annual reduction | 6% | 23% | 31% | 45% | 12% | 29% |

### Difficulty and Progress

Table 4 shows improvements for quicker and slower problem types separately. Figures 11 and 10 show the relationship between initial problem difficulty and progress for individual problem types. For the 25th percentile problems, the previous trend is clear: Problems that took longer to begin with saw more improvement. Problems that initially took less than a second virtually all became slower (see figure 10), for an average improvement of $-25\%$. The longest problems became 76% quicker on average, and none took longer than before. At the 75th percentile, there is no such pattern.

Table 4: Improvements by problem difficulty. Buckets contain problems of different lengths ($T_0$ denotes the earlier time). For each bucket and time period, the table shows (1) the mean fraction of time taken in the second competition compared to the first competition, (2) the number of problems in that class, and (3) the annual time reduction for that bucket and period.

| | Bucket 1 ($T_0 < 1$s) | | | Bucket 2 (1s $< T_0 <$ 100s) | | | Bucket 3 (100s $< T_0$) | | |
|---|---|---|---|---|---|---|---|---|---|
| 25th percentile | 1.57 | 10 | $-25\%$ | 0.45 | 10 | 33% | 0.12 | 7 | 76% |
| 75th percentile | | 0 | | 0.28 | 4 | 47% | 0.57 | 14 | 25% |

## 4.2. João Marques-Silva's Records

Table 5 shows a collection of historical times to solve SAT instances reported by João Marques-Silva. The instances were selected to illustrate the fast progress made in SAT solvers, in particular those targeting practical and industrial problems. The hardware used is unclear. Marques-Silva later compared a set of algorithms from different times on nine benchmark problems (Marques-Silva 2010, 59). The results of this are shown in table 6. Figure 12 shows both of these sets of data graphically.

In this data, the largest speedups tend to be earlier. In particular, SAT solvers seem to jump from timing out on an instance to solving it in a modest amount of time, representing much greater improvement than in other periods.

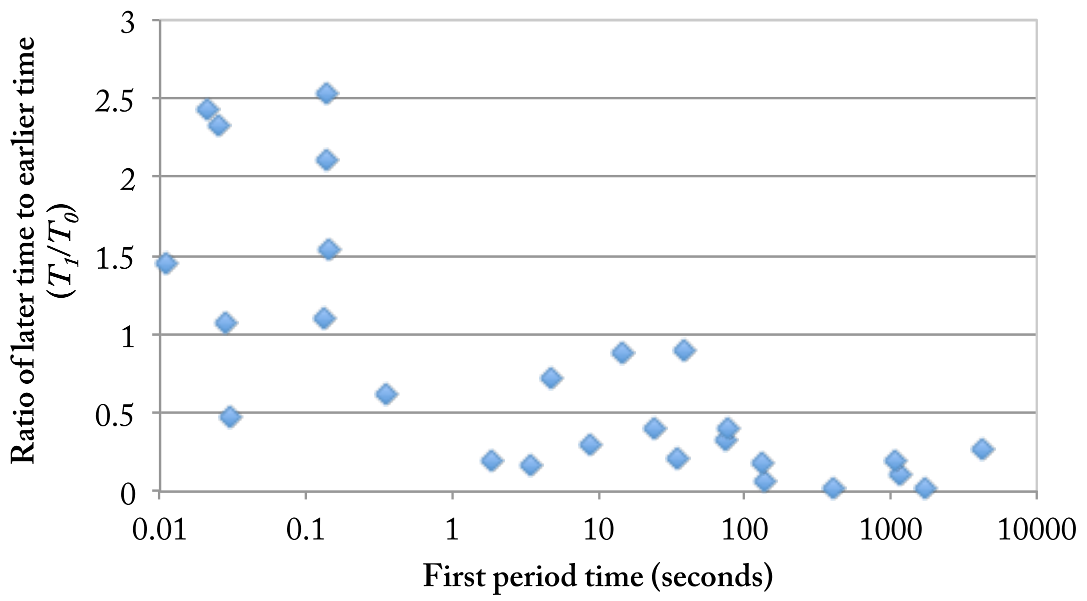


Figure 10: Initial difficulty and fractional improvement over two years for each random problem type, using 25th percentile times. The horizontal axis shows how long the problem initially took ($T_0$); the vertical axis shows the ratio of its time in the second competition ($T_1$) to its time in the first competition.

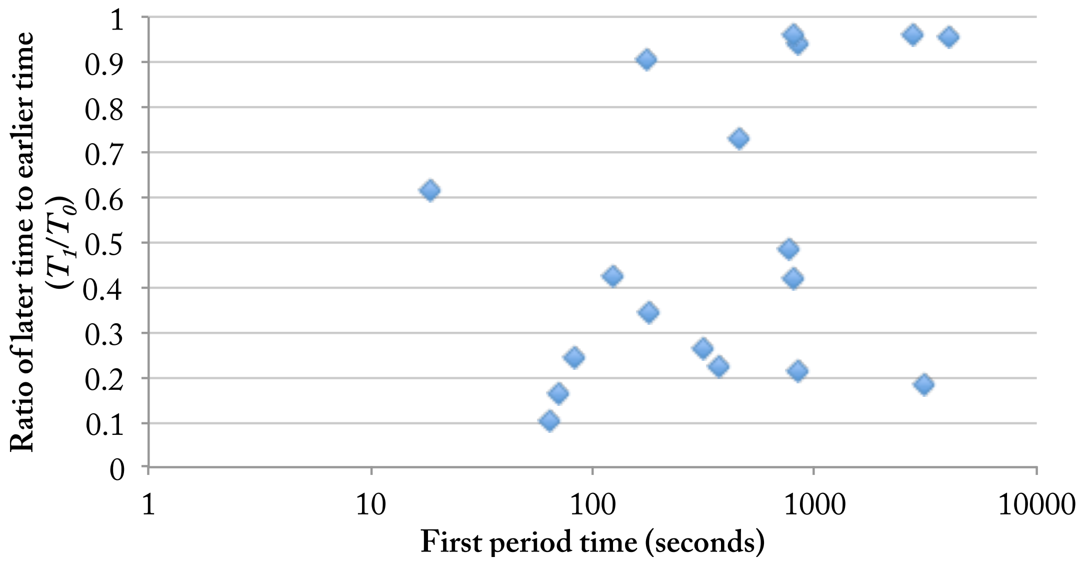


Figure 11: Initial difficulty and fractional improvement over two years for each random problem type, using 75th percentile times. The horizontal axis shows how long the problem initially took ($T_0$); the vertical axis shows the ratio of its time in the second competition ($T_1$) to its time in the first competition.

Table 5: Historical times, in seconds, to solve assorted SAT instances, collected by João Marques-Silva.

| Instance | 1994 | 1996 | 1998 | 2001 |
|---|---|---|---|---|
| ssa2670-136 | 40.66 | 1.2 | 0.95 | 0.02 |
| bf1355-638 | 1805.21 | 0.11 | 0.04 | 0.01 |
| pret150_25 | >3000 | 0.21 | 0.09 | 0.01 |
| dubois100 | >3000 | 11.85 | 0.08 | 0.01 |
| aim200-2_0-no-1 | >3000 | 0.01 | 0 | 0 |
| 2dlx__bug005 | >3000 | >3000 | >3000 | 2.9 |
| c6288 | >3000 | >3000 | >3000 | >3000 |

Table 6: Times, in seconds, to solve assorted SAT instances using different historical solvers (Marques-Silva 2010, 59). It is unclear whether the hardware used was historical or contemporary.

| Instance | Posit '94 | Grasp '96 | Chaff '03 | Minisat '03 | Picosat '08 |
|---|---|---|---|---|---|
| ssa2670-136 | 13.57 | 0.22 | 0.02 | 0.00 | 0.01 |
| bf1355-638 | 310.93 | 0.02 | 0.02 | 0.00 | 0.03 |
| design_3 | >1800 | 3.93 | 0.18 | 0.17 | 0.93 |
| design_1 | >1800 | 34.55 | 0.35 | 0.11 | 0.68 |
| 4pipe_4_ooo | >1800 | >1800 | 17.47 | 110.97 | 44.95 |
| fifo8_300 | >1800 | >1800 | 348.5 | 53.66 | 39.31 |
| w08_15 | >1800 | >1800 | >1800 | 99.1 | 71.89 |
| 9pipe_9_ooo | >1800 | >1800 | >1800 | >1800 | >1800 |
| c6288 | >1800 | >1800 | >1800 | >1800 | >1800 |

Three of the benchmarks in these datasets overlap, and the times listed for two of them in overlapping years are different. This may be from using different top-of-the-line algorithms for those years, and should give some idea of the variation in measurement. In figure 12, the lower number is used for each.

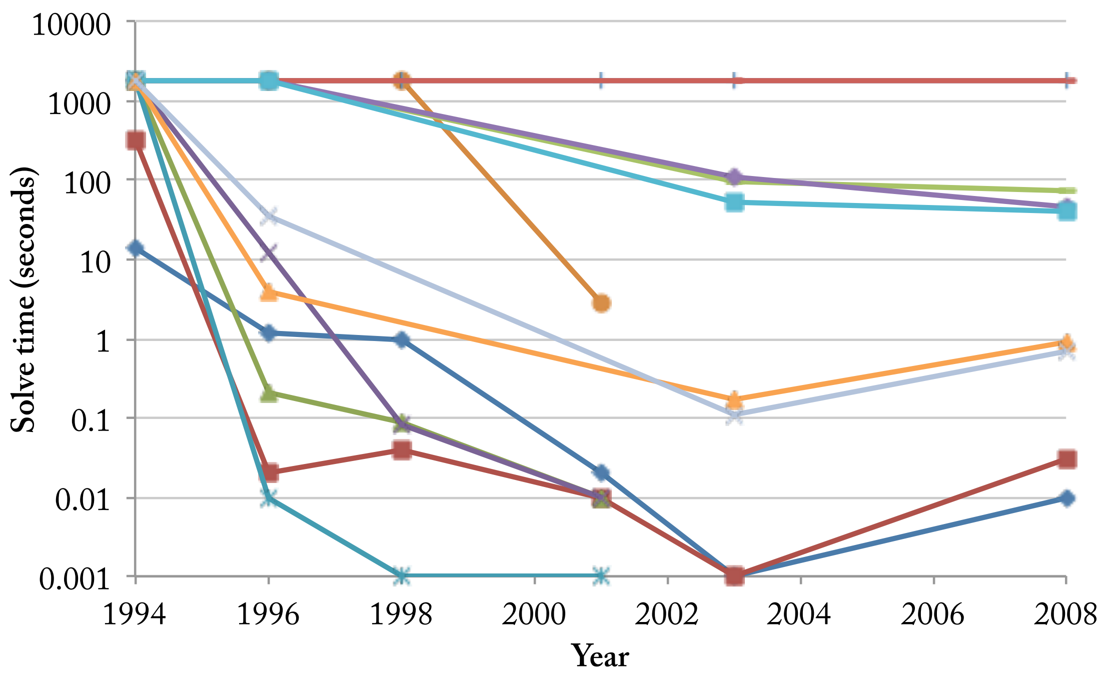


Figure 12: Solve times for historical state-of-the-art SAT instances (combined data from tables 5 and 6). Note that values of 1,800 represent a timeout and do not mean that the problem was solved in that time. Times listed as zero are entered as 0.001. Each line represents a different instance.

## 5. Game Playing

### 5.1. Chess

#### 5.1.1. The Effect of Hardware

To interpret historical comparisons between chess programs, it is useful to first understand the effect of hardware on performance. Levy and Newborn (1991, 192) estimated that a sixfold increase in speed allows a chess engine to search one level deeper in a tree of moves. The effects of hardware on performance, however, are somewhat confusing. Levy and Newborn report that in the 1970s, Thompson played versions of BELLE against each other in an attempt to factor out all contributors to performance other than speed. He found around a 200-point difference for each level searched between 1,400 and 2,000 Elo points. Above 2,000 Elo and seven levels, improvement per level slowed. Newborn did a similar study over further levels and found a similar slowing of Elo gains from searching more levels. However, the speed of this decline is not reported. For reference, 2,000 Elo is around the level of the best chess programs in 1980.

Levy and Newborn try to further substantiate the estimate of 200 Elo per level searched by showing that historically each additional level searched due to hardware improvement between 1978 and 1989 corresponded to a 200-point increase in Elo rating of the best programs. This would require roughly zero software progress between 1978 and 1989, however, which seems implausible. Also, this period is largely after chess programs had Elo ratings over 2,000, so the earlier data would predict smaller gains to hardware.

To confirm that substantial software progress does occur, we can look at the CCRL (2013) comparison of Rybka engines. Rybka 1.1 64-bit was the best of its time (the year 2006), but on equivalent hardware to Rybka 4.1 it is rated 204 points worse (2,892 vs. 3,102), and on the main CCRL list we can see that Komodo CCT 64-bit—the highest-rated engine using the same number of CPUs—is 256 points better than Rybka 1.1. Virtually all of the progress since 2006 seems to have occurred in one jump (see figures 13, 14, 15, and 17), so rates of software progress in recent times are not obviously generalizable to the longer term, where progress appears to have been steadier. However, given the current contribution of software to progress, it would be surprising if hardware accounted for virtually all progress in the past.

SSDF changed their hardware in 2008, doubling the processor speed, quadrupling the number of cores, and moving from a 32-bit to a 64-bit operating system. This produced an average rating increase of 120 points. The CCRL comparison of Rybka engines suggests doubling the number of machines gives 30–60 Elo points, and moving from a 32- to 64-bit operating system gives around 30 points. This leaves around 0–30 Elo points of the SSDF change to attribute to doubling processor speed. These numbers are close to those found in forum folklore: 60 points for doubling processor speed and 40 points for doubling processors.

Go improves with hardware to a similar degree to that suggested here for chess. A factor of six speedup gives around 260 Elo points in the middle of the range, but this declines considerably (see figures 20 and 21).

### 5.1.2. Computer Chess Ratings List (CCRL)

CCRL is one of several chess engine ratings lists. The Swedish Chess Computer Association (SSDF) and Chess Engine Grand Tournament (CEGT) below are others. Figure 13 shows the top ratings on versions of their list, archived since 2006. Hardware and time are equivalent to 40 moves in 40 minutes on Athlon 64 X2 (2.4 GHz) (4,600+ rating since 18 January 2007, and 3,800+ before). The two large jumps in 2006 and 2008 correspond to changing versions of Rybka, though twelve other program changes are not so visible (see red dots in the figure). Graham Banks (2013) of the CCRL project says the jumps toward the end are most likely the result of the CCRL's experiments with adjusting their rating methods.

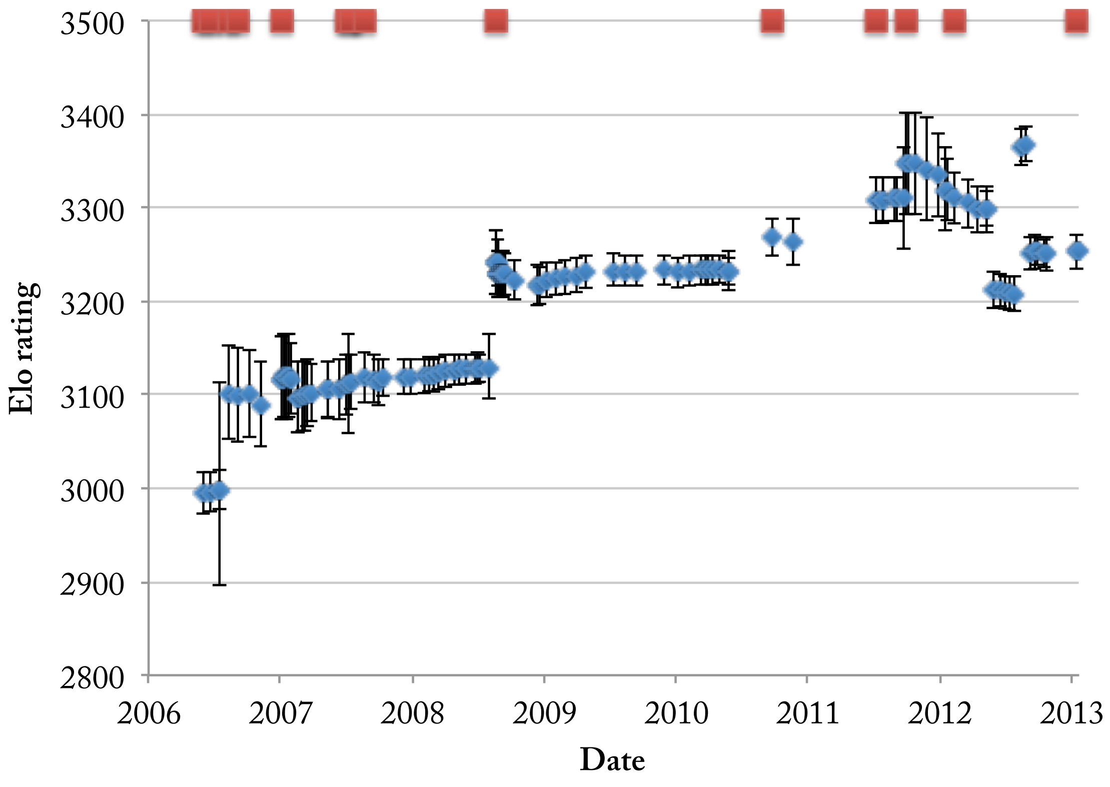


Figure 13: Elo ratings of the top chess-playing programs since 2006, with 95% confidence intervals. Red dots indicate new engines taking the lead. Data from CCRL (2006–2013).

### 5.1.3. Swedish Chess Computer Association (SSDF) Ratings

#### Archived Versions

Figure 14 shows Elo ratings of the top chess-playing algorithms over time on the SSDF site (available at the Internet Archive). This data covers roughly the same period as the CCRL data above but paints a strikingly different picture: all of the progress is made in one jump around the end of 2008. The jump perfectly corresponds to moving from all programs running on an Arena 256 MB Athlon 1200 MHz to some programs running on a 2 GB Q6600 2.4 GHz computer, suggesting the change in hardware accounts for the observed improvement. However, it also corresponds perfectly to Deep Rybka 3 overtaking Rybka 2.3.1. This latter event corresponds to huge jumps in the CCRL and CEGT records at around that time, and they did not change hardware then. The average program in the SSDF list gained 120 points at that time (Karlsson 2008), which is roughly the difference between the size of the jump in the SSDF records and the jump in records from other rating systems. So it appears that the SSDF introduced Rybka and new hardware at the same time, and both produced large jumps.

Other than the jump, ratings appear to be moving downward as often as upward. This makes little sense as a model of the best abilities, as older programs continue to exist.

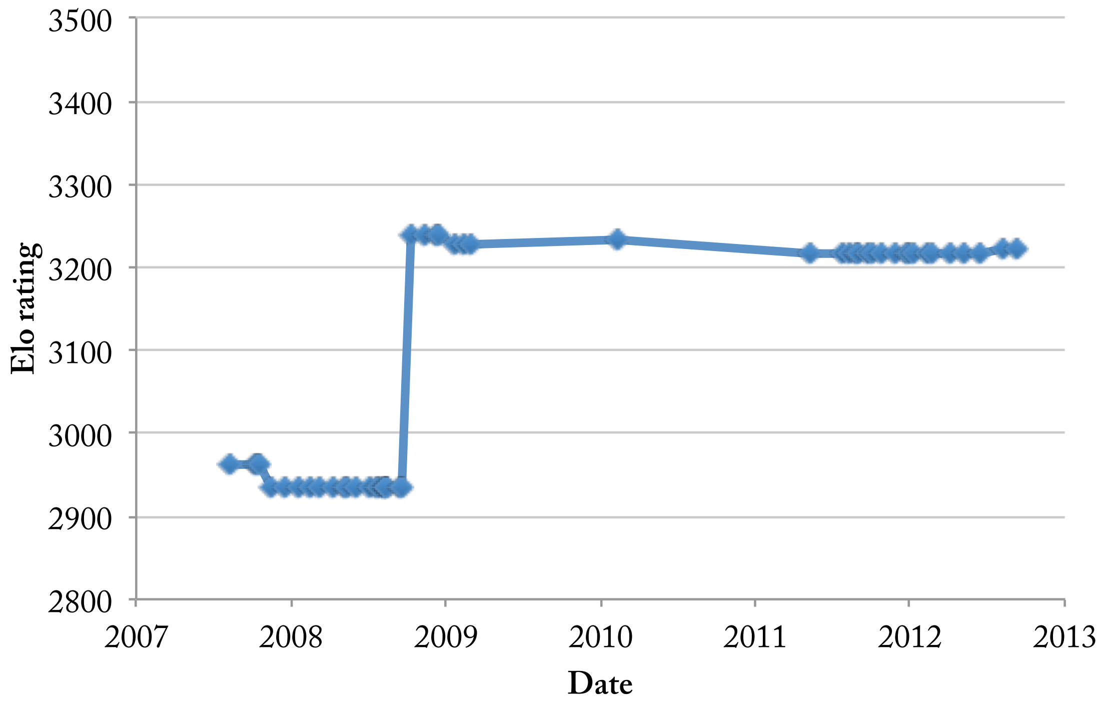


Figure 14: Elo ratings of the top chess-playing programs since 2007. Data from SSDF (2007–2012).

### Wikipedia Records

Wikipedia maintains a list of "year-end leaders," illustrated in figure 15. Between 1985 and 2007, progress was fairly smooth, at 50 Elo per year.

### SSDF via ChessBase 2003

Figure 16 is taken from a 2003 article by Jeff Sonas, also apparently from old SSDF data. It appears to be a little bumpier than the Wikipedia account.

### 5.1.4. Chess Engine Grand Tournament (CEGT) Records

Figure 17 shows ratings of programs at the top of CEGT's 40/4 ratings list over time, collected from archived versions. The large drop at the start of 2012 can't reflect reality, as all of the software available at the end of 2011 should still have been available in 2012. Furthermore, all of the top engines dropped their ratings by around 200 points at the same time in 2012, which suggests an adjustment of methods. The drop at first appears to compensate for a similarly sized jump at the end of 2008 but, as noted earlier,

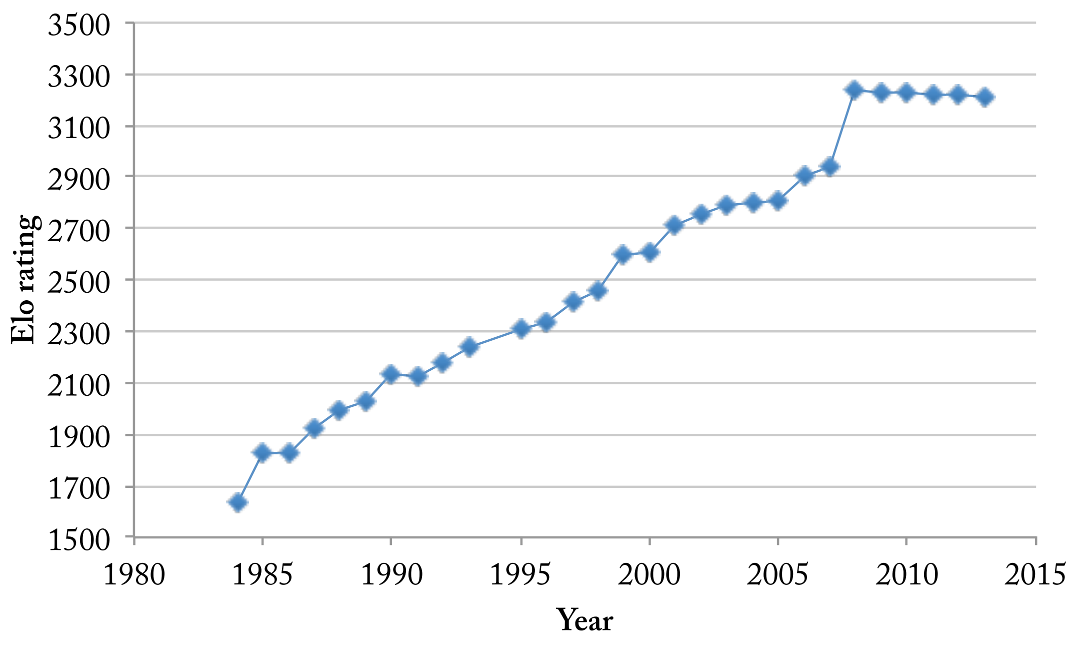


Figure 15: Elo ratings of the best program on SSDF at the end of each year. Data from Wikipedia (2013).

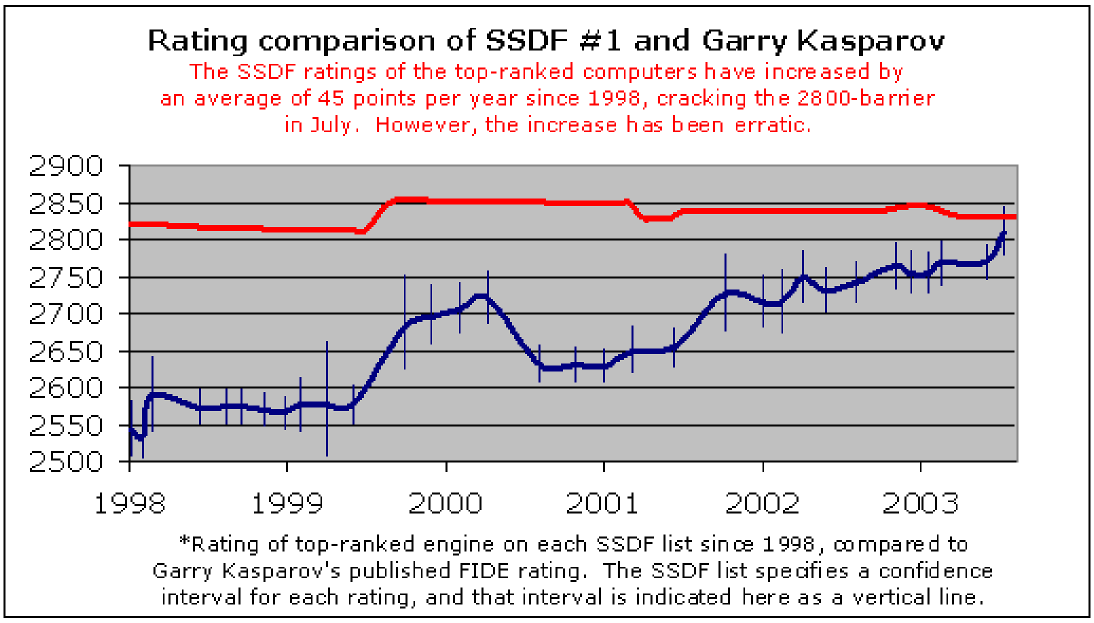


Figure 16: Sonas's account of SSDF rankings. Reprinted from Sonas (2003).

that jump coincides with jumps in CCRL and SSDF data, which suggests it reflects something real.

### 5.1.5. Dr Dobb's Article Estimates

In 1994, L. Stephen Coles produced figure 18, and calculated from it a roughly forty-point per year growth rate in best computer Elo ratings since 1960.

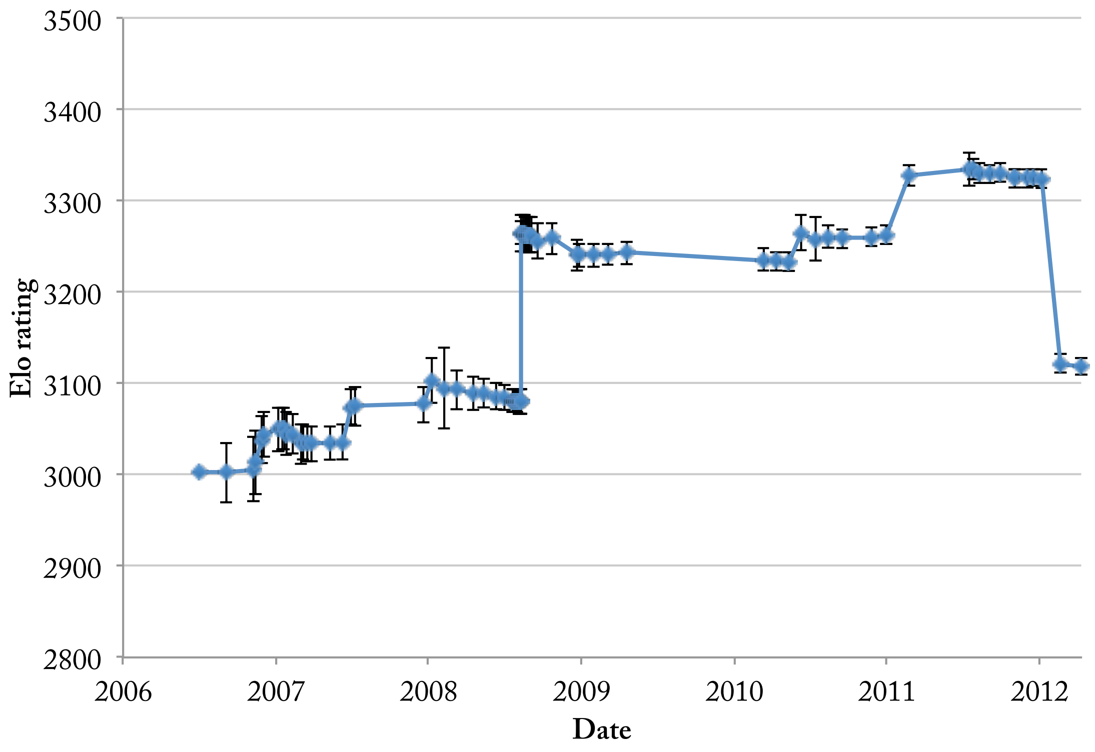


Figure 17: Top engine ratings over time with data from the CEGT (2006–2012) 40/4 ratings list.

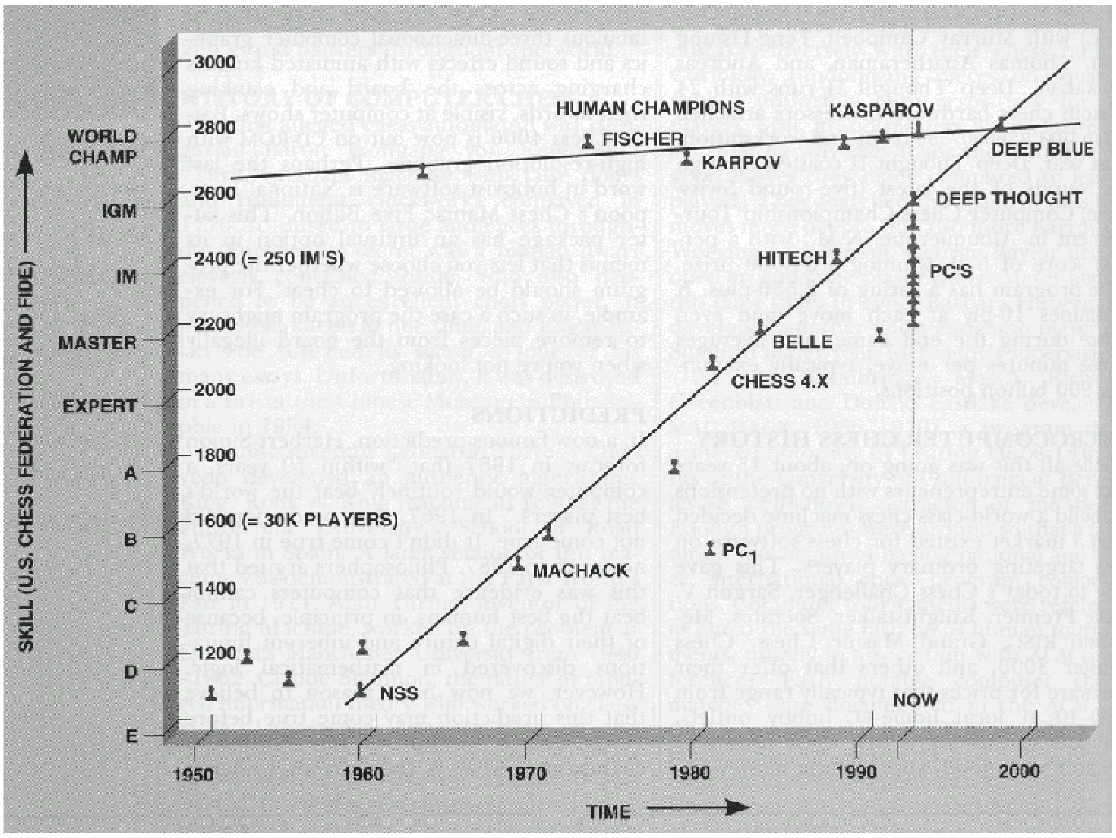


Figure 18: Early chess progress according to L. Stephen Coles. Reprinted from Coles (2002, fig. 1).

### 5.1.6. Historical Estimates for Well-Known Chess Programs

Figure 19 shows some estimates of early chess ratings from a variety of sources compiled in 2011 by Luke Muehlhauser, who comments:

> Page 71 of Hans Moravec's *Robot* (1998) is my source for Year and ELO data on MacHack (1970), Chess 2.5, Chess 4.0, Chess 4.9, Cray Blitz, HiTech, Chiptest, Deep Thought, Deep Thought II, and Deep Blue. *Computer Museum Report, Vol. 20* is my source for Year and ELO data on MacHack (1965), Chess 3.0, and Chess 4.6. The Wikipedia entry for "Swedish Chess Computer Association" (accessed 10-09-2011) is my source for Fritz 6.0, Chess Tiger 14.0, Deep Fritz 7.0, Shredder 8.0, Rybka 1.2, and Deep Rybka 3. I excluded entries on these lists from before 1965 and after 2008. I also removed a few entries to avoid clutter; these entries were not outliers. (Muehlhauser 2011)

The later years use the Wikipedia SSDF list shown above, but many of the earlier points are new.

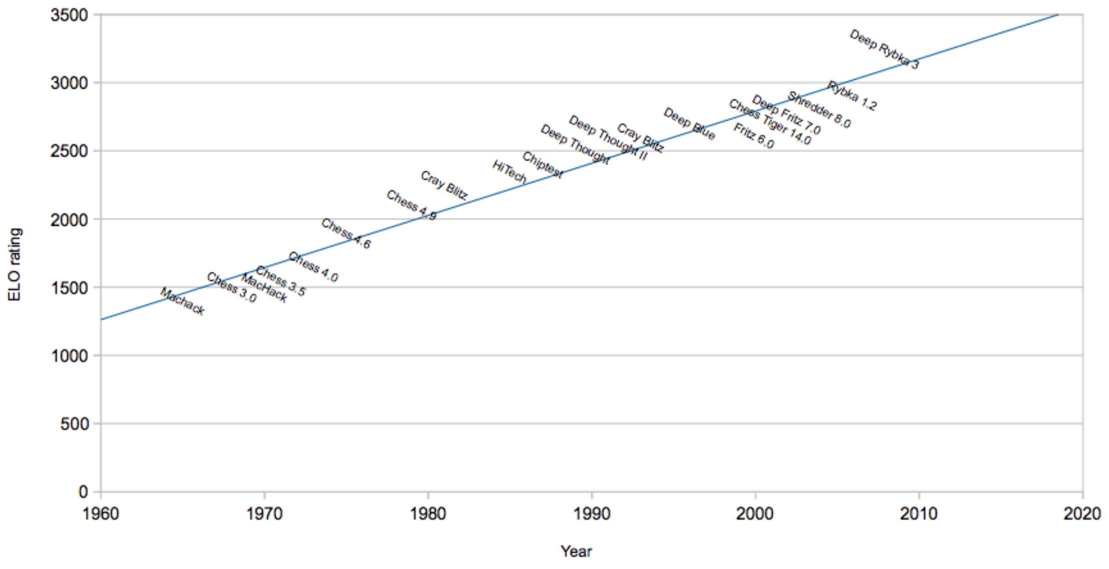


Figure 19: Historical chess engines' estimated Elo ratings from a variety of sources. Reprinted from Muehlhauser (2011).

### 5.1.7. A Note on Open-Source Engines

It is often claimed that good open-source engines heralded a burst of progress in computer chess, due to sharing of insights. Verifying and quantifying such returns from sharing would be interesting from the perspective of predicting the future of AI, in particular the relative likelihoods of fast, local takeoff scenarios relative to slower, more global ones.

Some people in the field say that open source programs have been very helpful. For instance, Houdini is currently the top-rated chess program. Here are excerpts from an interview with Robert Houdart, its author:

> I started with this idea to build the best chess engine that I could—and I was helped a lot by the open culture that has come with the Internet. You know, two decades ago you had to invent every part of a chess engine from zero (and I've done my fair share of that), but today we're in a situation where techniques, ideas and examples are readily available on the Internet. You can call it a coming of age of the computer chess scene—as an engine author you're no longer obliged to sit in your corner reinventing the wheel. The computer chess Wikipedia, some strong open source engines, and discussions on Internet forums about chess programming techniques and ideas make the design and development of a strong engine a lot easier than, say, twenty years ago. . . .
>
> *How much do you owe to other programs and programmers? Did you collaborate with anybody, did you receive any advice and assistance?*
>
> As I mentioned earlier, the Internet community is a great source of inspiration and the information that is now available in seconds would have taken ages to collect twenty years ago. Other than the Computer Chess Wiki, which is an awesome resource for any aspiring chess engine developer, I must credit the Stockfish open source engine, which was the inspiration for the multithreaded implementation of Houdini, and the IPPOLIT open source engine that provided a whole array of search and evaluation techniques. The development effort is done entirely by myself, but I'm supported by people from around the world that send ideas for improvement, very often positions in which Houdini doesn't perform well. Some fans even have donated hardware for engine testing. It's amazing how supportive the community has been over the past two years. (ChessBase 2013)

The front page for the Houdini engine website also credits open-source projects for much of its strength:

> Without many ideas and techniques from the open source chess engines IPPOLIT and Stockfish, Houdini would not nearly be as strong as it is now. (Houdart 2012)

However, the ratings data we have appears to show something like steady progress for many decades, a large jump in 2008 at the start of Deep Rybka 3, and then no progress since. Regardless of when we think open source came to fruition, there doesn't seem to be recent improved general progress to attribute to it.

### 5.2. Go

#### 5.2.1. Hardware

In order to make use of comparisons between Go programs, many of which have used different hardware, we need to understand the effect of hardware on performance. This is important because, for example, much of the available data on Go progress is from the KGS Go Server, and programs that play on the KGS Go Server may be run on arbitrary hardware and tend to use more impressive hardware for more important competitions.

Figure 20 is taken from a conference paper by Baudiš and Gailly (2011). The black circles show the strength of the Go program Pachi as the number of Monte Carlo simulations ("playouts") Pachi is allowed to complete per move grows (Elo rating was measured automatically against another Go program). The number of playouts is a very good proxy for computational intensity. The colored curves represent similar experiments on distributed clusters without fast, shared memory. These scale less well. In all cases, iterated doublings of computational intensity result in diminishing Elo gains.

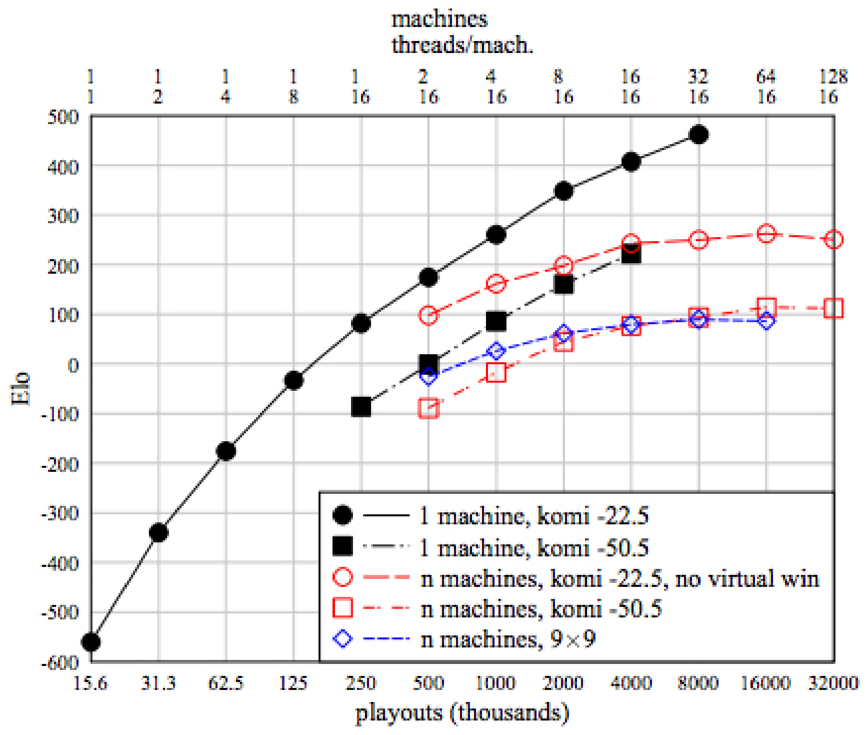


Figure 20: Go Elo improvements with hardware. Reprinted from Baudiš and Gailly (2011, fig. 4).

A 2008 project has very similar findings for two further Go programs, pictured below in figure 21. MoGo was a top-of-the-range program at the time, whereas FatMan was much weaker. Their initial hardware was adjusted to make them similarly strong (64 playouts for MoGo to 1,024 for FatMan), with each level indicated on the horizontal axis representing a doubling of computations. Again, doubling hardware adds a diminishing number of Elo points in the region of 100 per doubling.

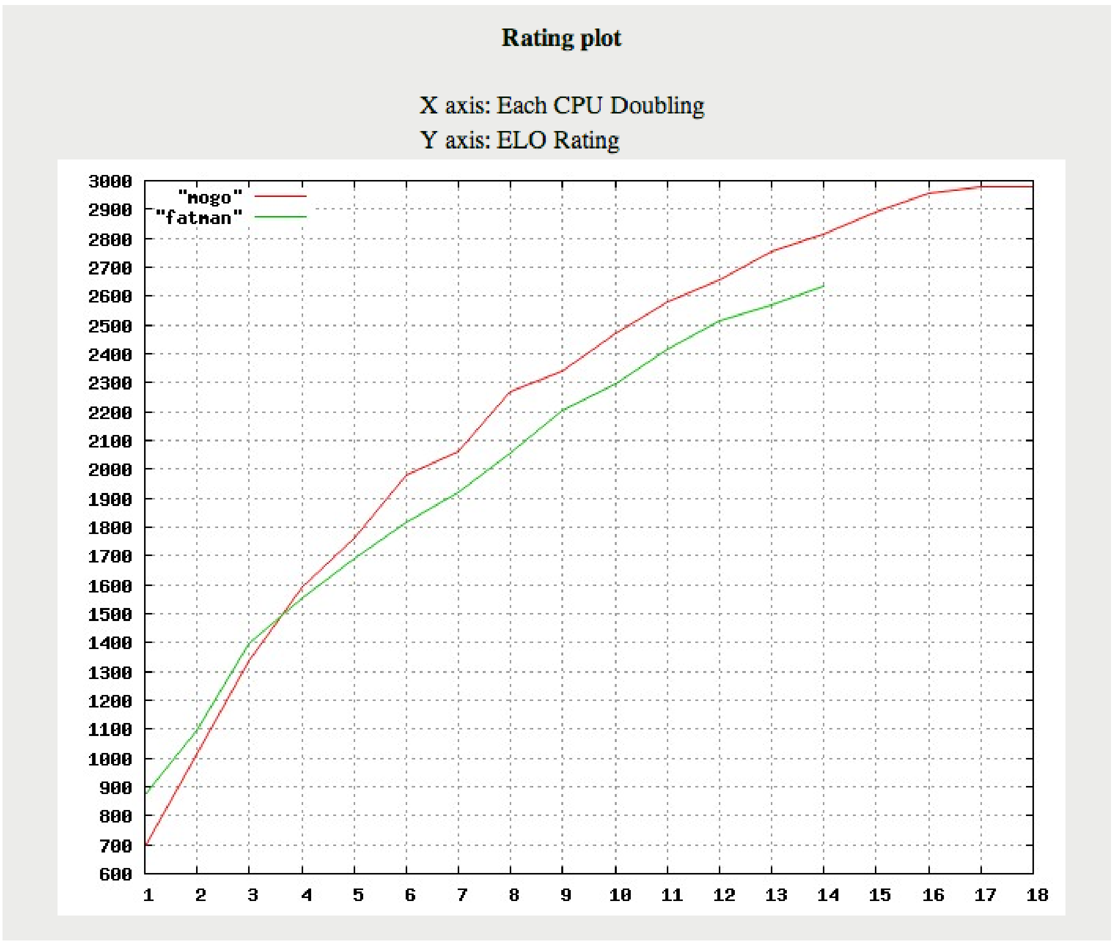


Figure 21: Go Elo improvements with hardware for a strong (red) and weak (green) program. Reprinted from Dailey (2008).

Relative Elo ratings correspond to probability of the better player winning. Figure 22 shows this correspondence. Around the middle of the spectrum, a Go program has roughly a 65% chance of winning against the same program with half the computing power.

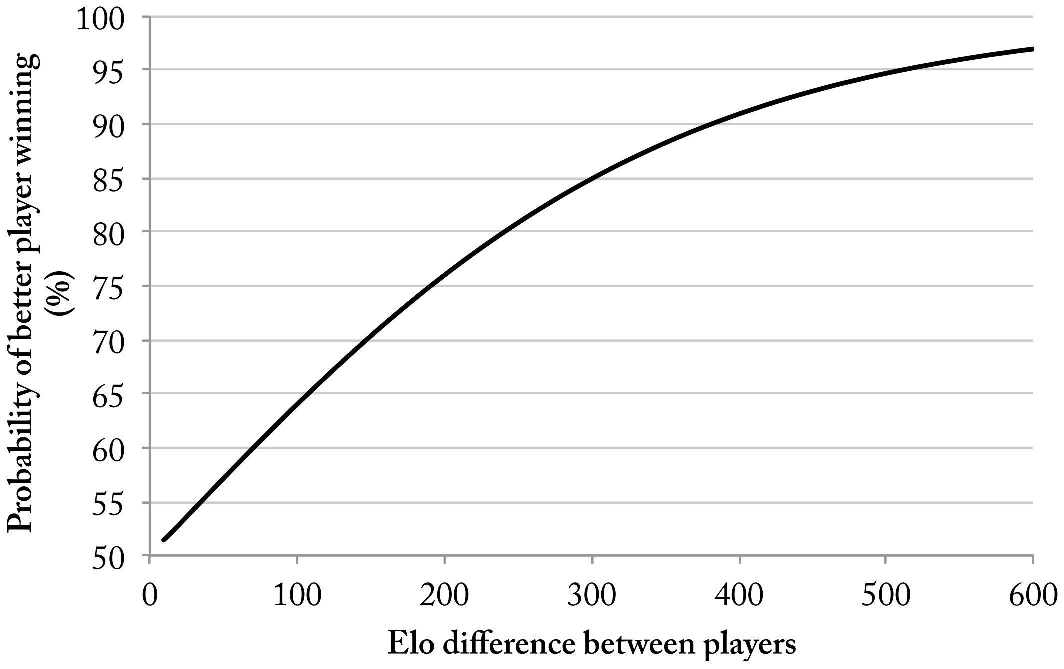


Figure 22: The Elo rating system: the probability of the better player winning versus the difference in Elo ratings.

### 5.2.2. Efforts

It appears that most Go programs are developed by individuals, pairs, or university-based teams. For instance Zen, Crazy Stone, Pachi, Steenvreter, Aya, Leela, MyGoFriend, Valkyria, and The Many Faces of Go are developed by individuals or pairs. Fuego, Nomitan, Orego, Gommora, and MoGo are developed by university groups.

### 5.2.3. Ing Challenges

The Ing Cup was a computer Go competition that ran until 2000. It included a challenge round where the winner of the cup was played against human "insei"—early teens with ratings of about 6d (European) (Wedd 2013). There were increasingly large prizes for programs winning under a series of increasingly small handicaps. Figure 23 shows challenge attempts by year, handicap level played for, and whether it was won (black) or lost (white). The programs should be among the best in the world at the time; the line of top computer Go-playing ability goes roughly over the white points and under the black points. Progress is a bit under a stone per year, but jumpy—in one year the record changed from seventeen stones to thirteen stones, while in another three years it didn't progress at all (though recorded progress had to be in two-stone increments). Data for hardware and time permitted are not readily available.

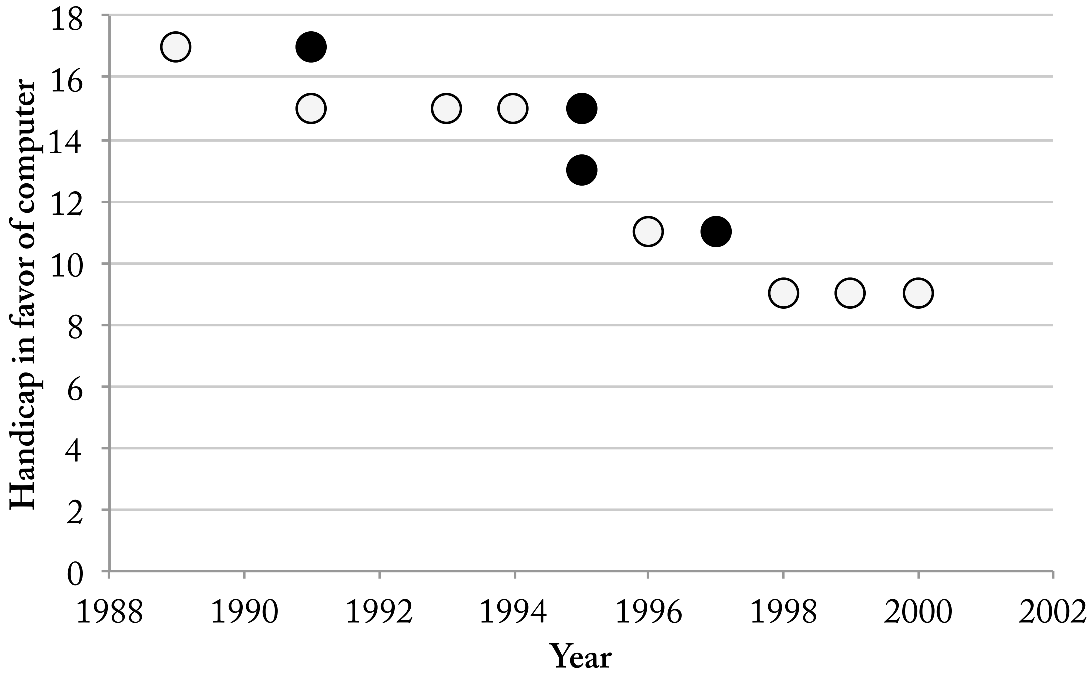


Figure 23: Successful (black) and unsuccessful (white) Ing Challenge attempts. Data from Wedd (2013) and Sensei's Library (2008).

### 5.2.4. Zen19 on KGS

Figure 24 shows progress of Zen19, a variant of the state-of-the-art Go bot Zen, since it began playing on KGS. The blue points are the mean of the program's score at the times it played tournament games each month. The score at that time is based on its interactions with other human users on the site, not just on those tournaments. The pink "×"s are median integer scores from inspecting some archived snippets of continuous KGS ratings data (median integers are reported because other measures weren't easily extracted). Progress is close to one dan per year.

### 5.2.5. Educated Guesswork

Another source of data is what members of the computer Go community have historically believed the standards of play have been. Figure 25 shows estimates from two sources: David Fotland—author of The Many Faces of Go, an Olympiad-winning Go program—and Sensei's Library, a collaborative Go wiki. David Fotland warns that the data from before bots played on KGS is poor, as programs tended not to play in human tournaments and so failed to get ratings. There seems to be general agreement that recent measures on KGS (such as those for Zen19 above) are a good measure of strength.

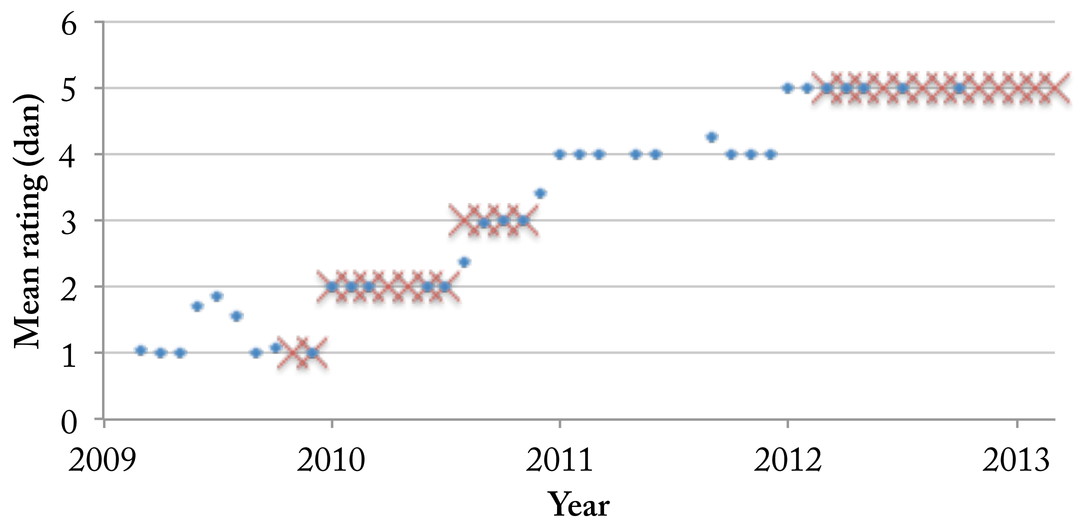


Figure 24: Zen19 ratings over time on KGS Go servers. Data from KGS (2010, 2013a).

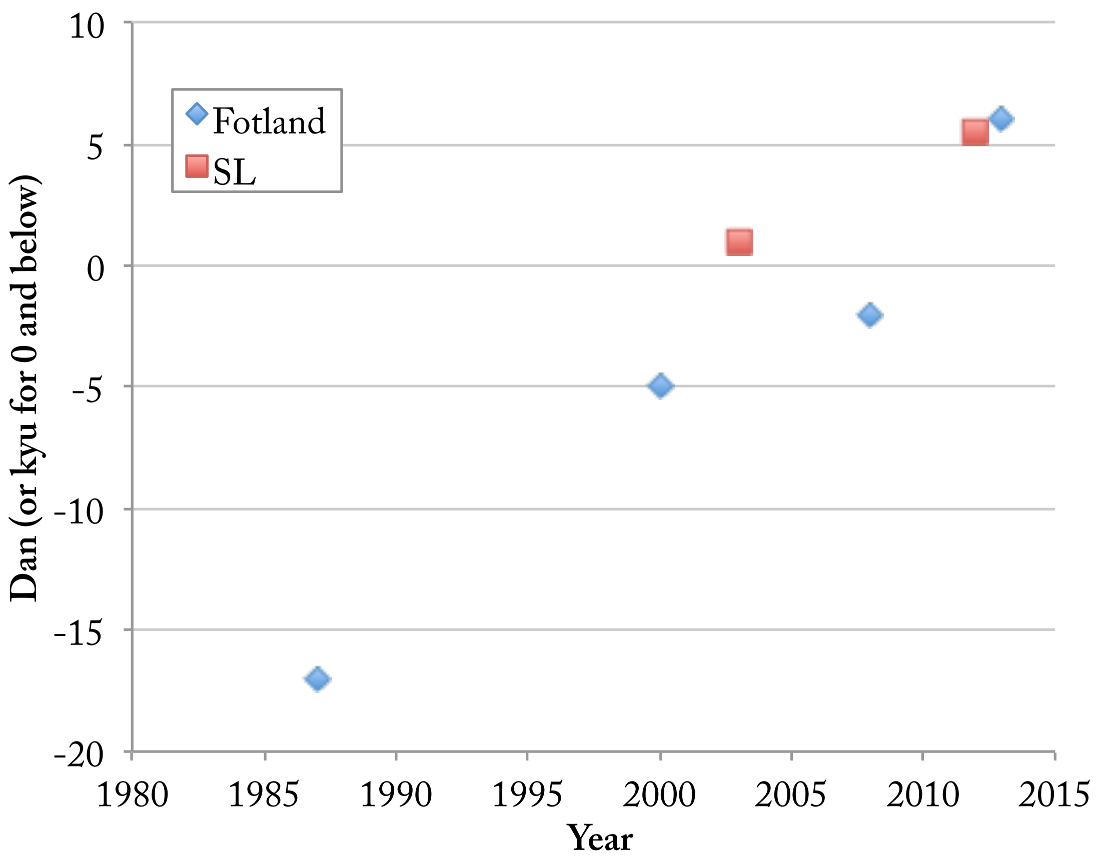


Figure 25: Rough estimates of historical Go bot abilities. Data from Sensei's Library (2013) and Fotland (2002).

### 5.2.6. KGS Tournament Data

When bots enter tournaments on KGS, their ratings are recorded. Figures 26, 27, and 28 show ratings of bots that entered three different KGS competitions since 2005. Ratings are determined by play on KGS, not just the tournaments in question. Many other

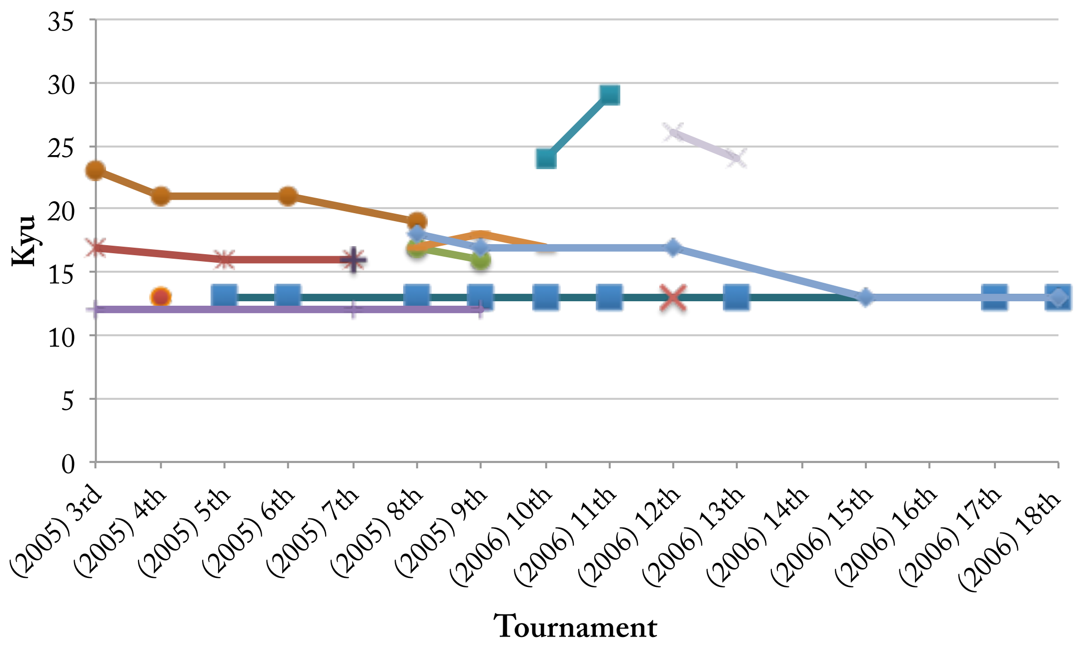


Figure 26: Go bot ratings from KGS computer Go tournaments in 2005 and 2006 (measured in kyu; note that larger numbers correspond to worse ratings). Each line represents a different Go bot. Data from KGS (2013b).

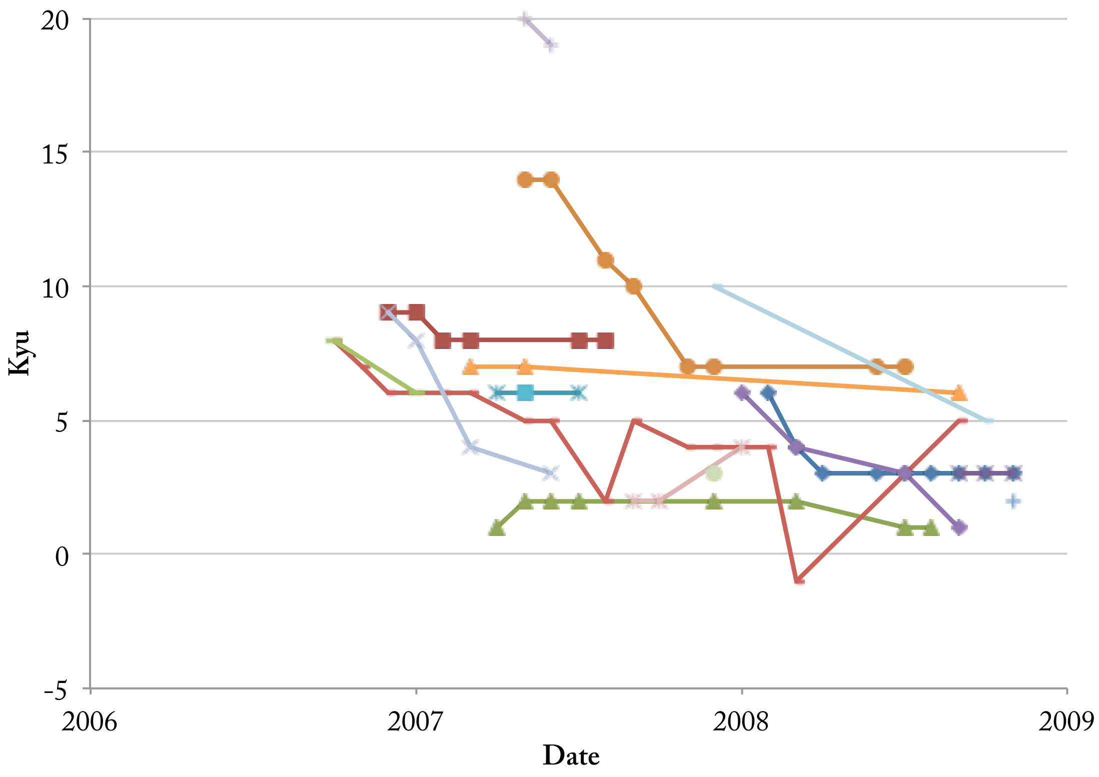


Figure 27: Go bot ratings from KGS bot tournaments formal division from 2006–2008 (measured in kyu; note that larger numbers correspond to worse ratings). Each line represents a different Go bot. Data from KGS (2013b).

bots entered these competitions without ratings, presumably due to not having played enough. Hardware was not fixed.

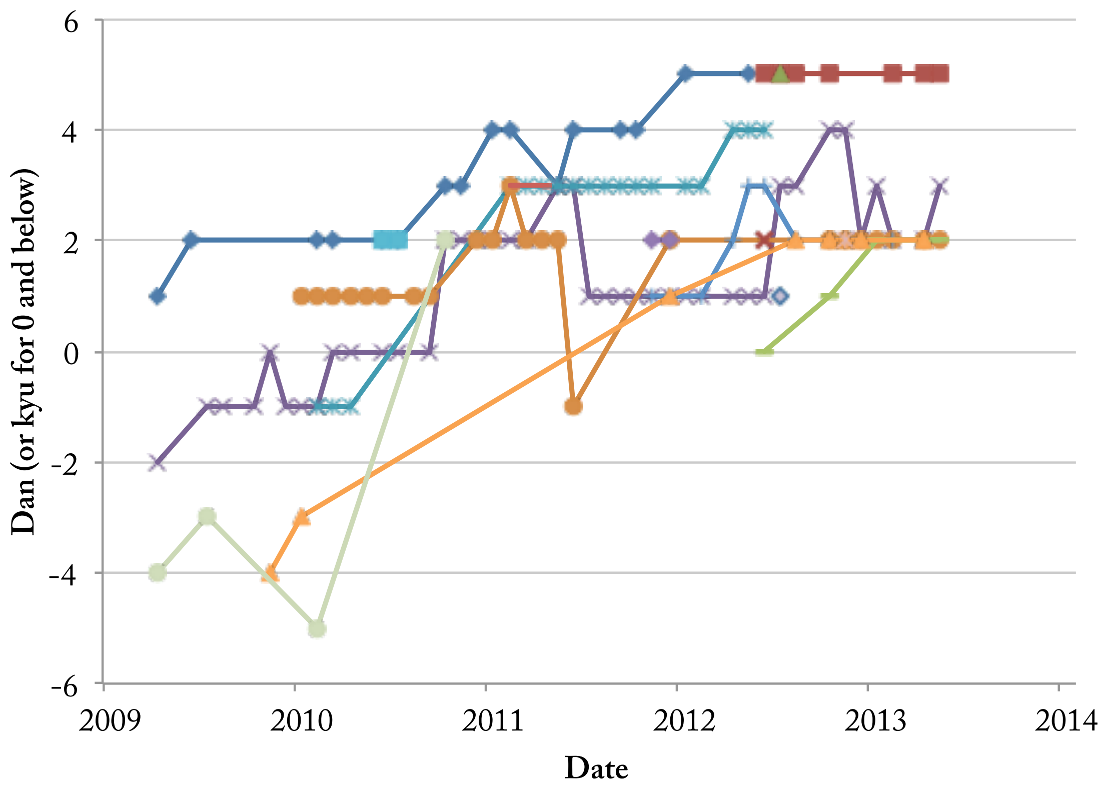


Figure 28: Go bot ratings from KGS bot tournaments 2009–2013 (measured in dan, treating kyu ratings as zero or below). Each line represents a different Go bot. This figure only includes bots that ever obtained a dan rating (positive on this graph), excluding one improbable data point. Data from KGS (2013b).

### 5.2.7. Sensei's Library

The KGS bot ratings page at Sensei's Library (2013) lists "a few KGS bots and when they achieved a rating and held it for at least 20 rated games in a row." These are probably obtained from KGS data in a different fashion from mine, so it is a reasonable sanity check for my method. Figure 29 confirms progress has been around a stone per year.

### 5.2.8. Computer–Human Contests

Nick Wedd (2013) maintains a list of human–computer Go matches at his website. The results are listed with the human's score first. They are somewhat hard to interpret, as one must account for the handicaps used, many of which are great.

## 6. Factoring

Figure 30 shows records for the factoring of large numbers. They come from data at FactorWorld and approximate interpretations of Carl Pomerance's A Tale of Two Sieves.

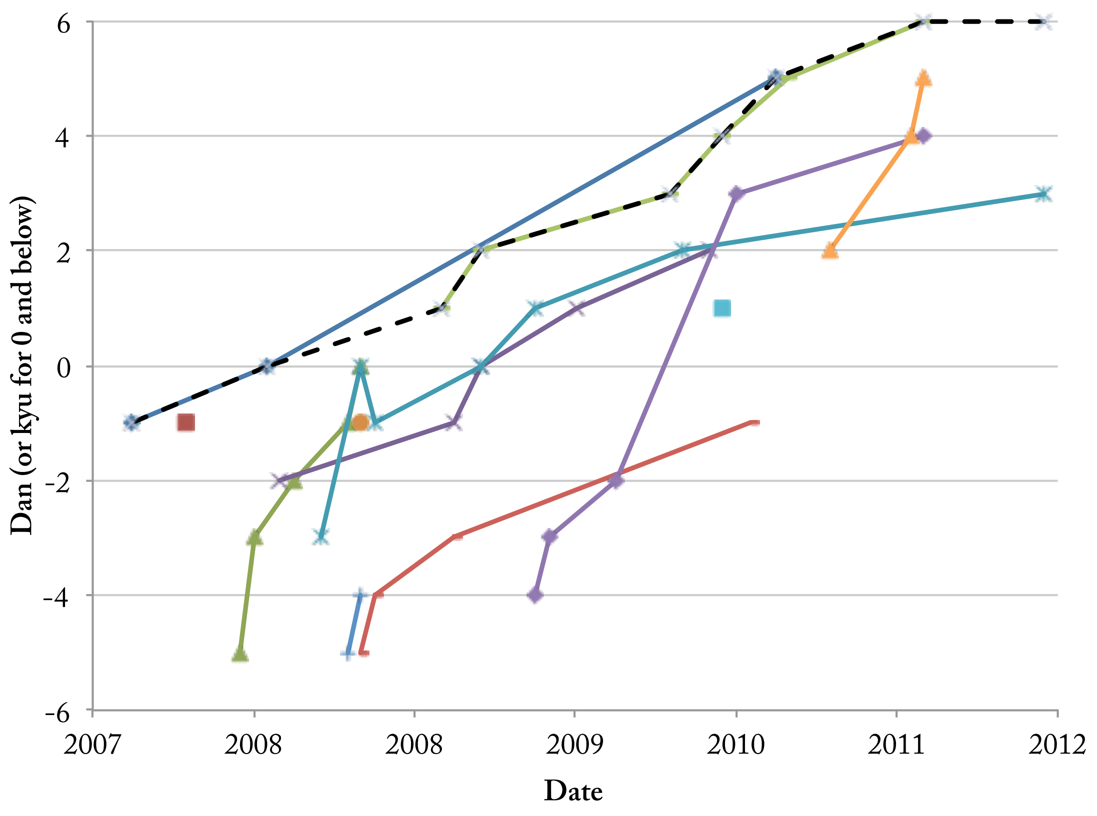


Figure 29: Progress of some highly rated Go bots on KGS Go Server (measured in dan, treating kyu ratings as zero or below). Each line corresponds to a different Go bot. The black dashed line represents the overall rating. Data from Sensei's Library (2013).

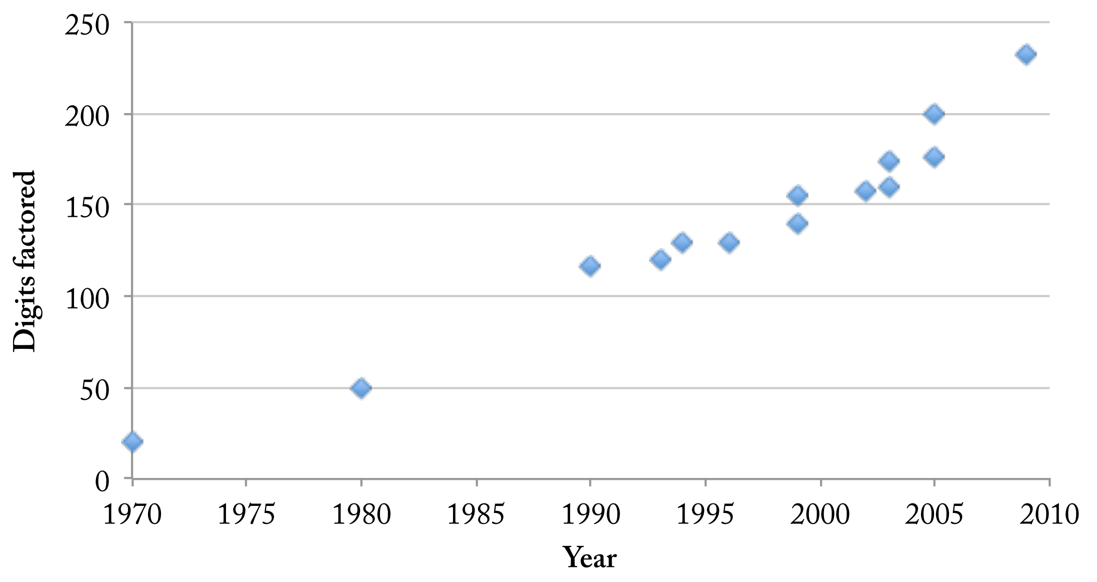


Figure 30: Factoring records by year and number of digits. Data from Contini (2010) and Pomerance (1996).

Many of the FactorWorld figures are originally from the RSA challenge, which ran from 1991 to 2007. Since the 1970s, the numbers that can be factored have apparently increased from around twenty digits to 222 digits, or 5.5 digits per year.

It is hard to find a good metric for progress in terms of returns on algorithmic improvements, since factoring of larger numbers has also involved much larger inputs of time, and there appear to have been no particular limits on time allowed for completion of such challenges. Ideally, we would like to know something like how the feasible number of digits has grown in fixed time. The scaling up of time spent, relative to that predicted by Moore's law making GHz-years quicker, suggests we have something like feasible digits in fixed sidereal time. An alternative would be to know how fast required time inputs are generally expected to grow with the number of digits, and measure progress against that expectation. But finding a reasonable expectation to use is difficult.

Solving times vary widely, and are generally much larger for later solutions. Figure 31 shows CPU times for the FactorWorld records. These times have increased by a factor of around ten thousand since 1990. At 2000 the data changes from being in MIPS-years to 1 GHz CPU-years. These aren't directly comparable. The figure uses 1 GHz CPU-year = 1,000 MIPS-years, because it is in the right ballpark and simple, and no estimates were forthcoming. The figure suggests that a GHz CPU-year is in fact worth a little more, given that the data seems to dip around 2000 with this conversion.

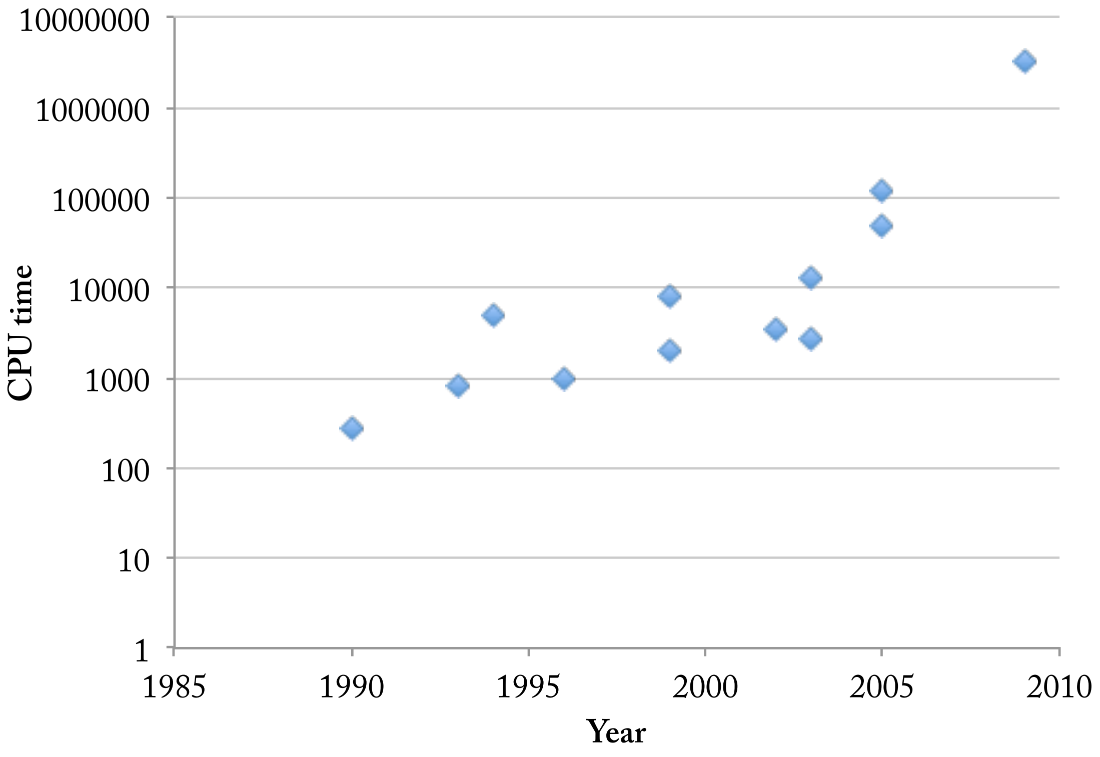


Figure 31: CPU time to factor numbers. Measured in 1,000 MIPS-years before 2000, and in GHz-years after 2000; note that these are not directly comparable, and the conversion here is approximate. Data from Contini (2010).

## 7. Physics Simulations

Figure 32, from "A Science-based Case for Large-Scale Simulation, Volume 2" (Keyes et al. 2004), shows progress in various scientific simulations. It is written with the apparent agenda of encouraging support for projects such as these scientific simulations, so the examples are likely selected to display progress.

It does not come with much explanation. Hardware improvements are shown as a straight blue line. The other lines appear to correspond to progress in somewhat different applications not attributable to hardware. It appears that effective gigaflops used by these simulations have increased exponentially, and nonhardware improvements have accounted for around as much of this as hardware improvements.

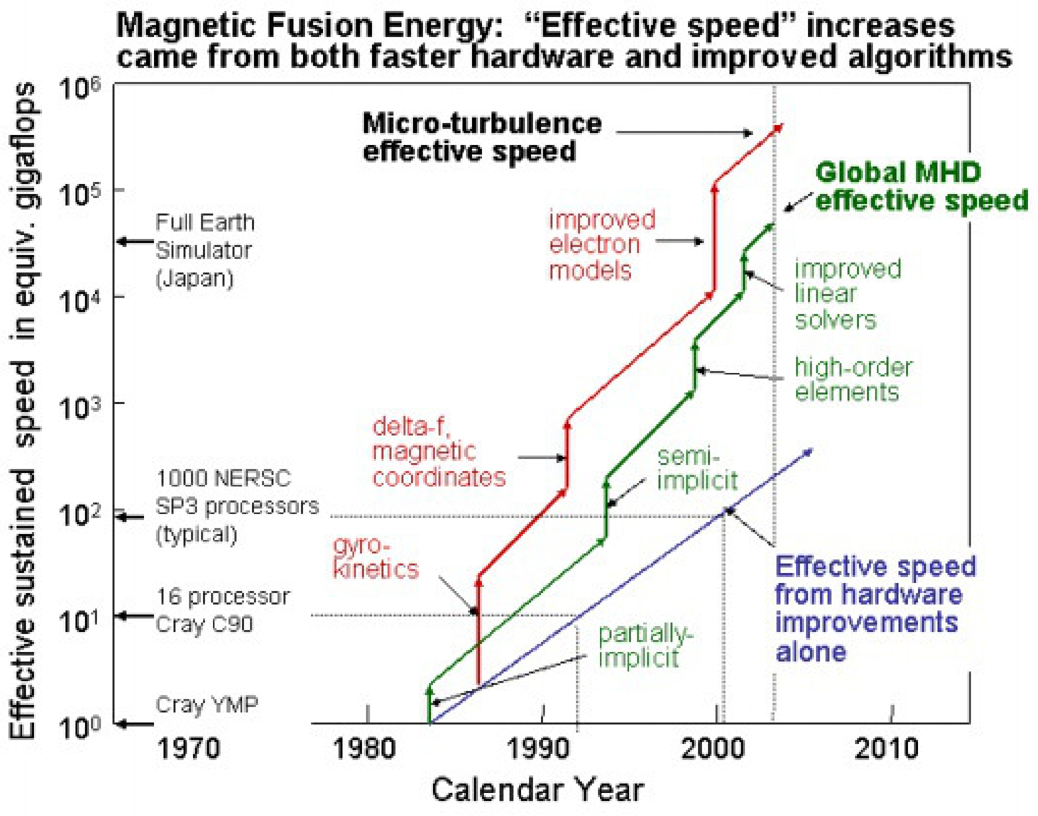


Figure 32: Effective sustained speed progress in magnetic fusion energy simulations from hardware and software improvements. Reprinted from Keyes et al. (2004).

Anecdotally, algorithmic advances have also greatly sped up simulating model spin systems:

> Progress in computational materials science and nanoscience has traditionally been advanced at least as much, and usually more, by breakthroughs in algorithms than by increases in computing power as David Landau has pointed out. Improvements in computer speed (Moore's law) alone would

account for three orders of magnitude increase in performance over the period 1970–1995, whereas theoretical insights and algorithmic advances have resulted in an additional seven orders of magnitude increase in the speed of Monte Carlo algorithms for simulating the model spin systems. (Keyes et al. 2004, 114)

## 8. Operations Research

### 8.1. Linear Optimization

#### 8.1.1. CPLEX

CPLEX is a mixed integer programming (MIP) solver. Versions of it were released between 1988 and 2008. Most of these versions (1.2 to 11.0, released in 1991 to 2007) were tested on a set of 1,852 MIP problems. The problems are “real-world” MIP problems, taken from a library of 2,791 after removing many considered too easy, a few considered too hard, and some duplicates (Bixby 2010). The details of the experiment suggest that this was done at one time, on fixed hardware. The speedups between versions are shown in figure 33.

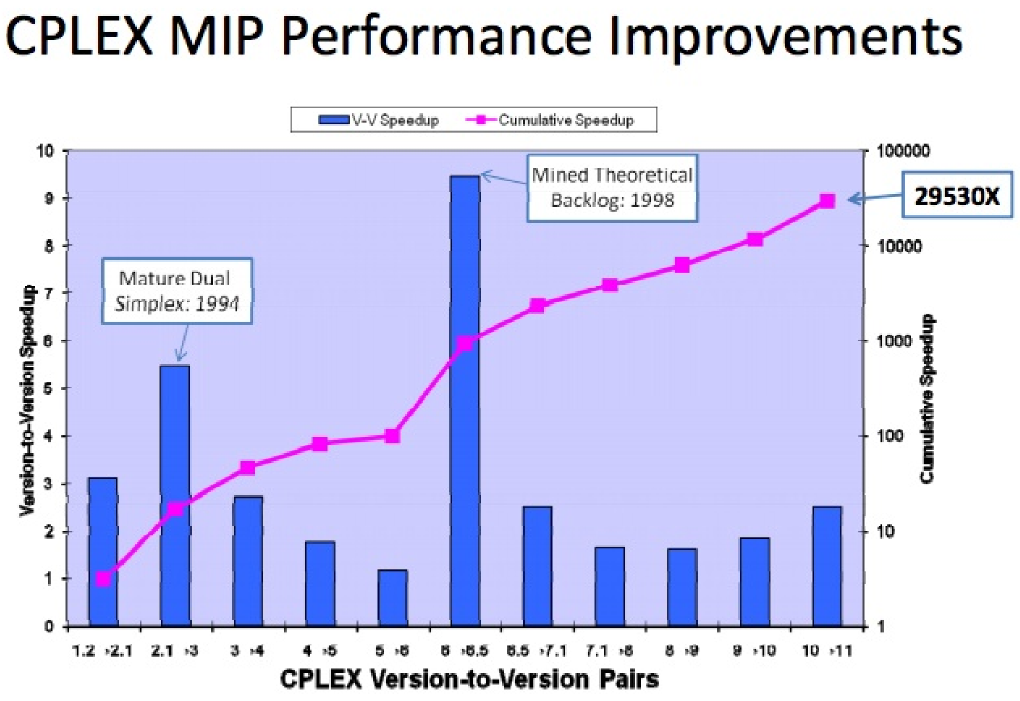
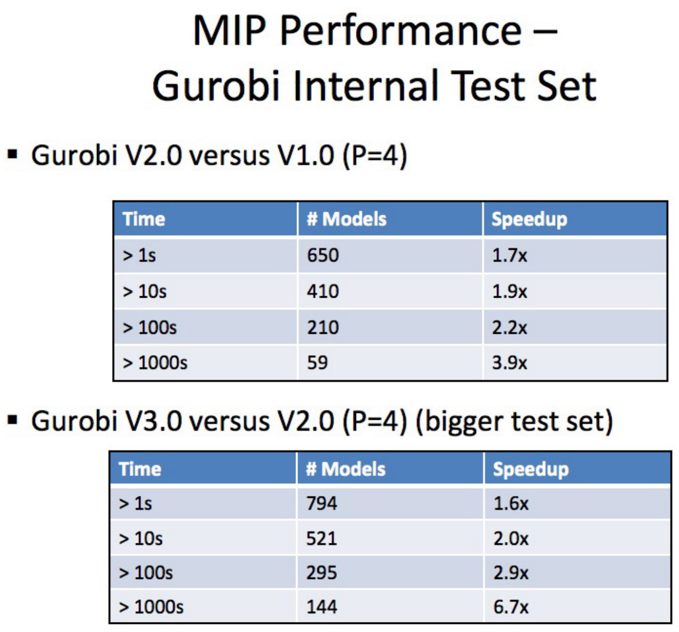


Figure 33: Speedup between versions of CPLEX on 1,852 MIP problems. Note that the horizontal axis is not to scale temporally, as versions were released at uneven intervals. Reprinted from Bixby (2010).

### 8.1.2. Gurobi Internal Testing

Gurobi is another MIP solver, developed in 2009. Figure 34 shows speedups in Gurobi for a range of problem sizes between version 1.0 (May 2009), version 2.0 (Oct 2009), and version 3.0 (April 2010), collected by Gurobi's authors (Bixby 2010). Gurobi improved its speed across a range of problem sizes by a smaller factor per time between versions 2.0 and 3.0 than 1.0 and 2.0.



MIP Performance –
Gurobi Internal Test Set

- Gurobi V2.0 versus V1.0 (P=4)

| Time | # Models | Speedup |
|---|---|---|
| > 1s | 650 | 1.7x |
| > 10s | 410 | 1.9x |
| > 100s | 210 | 2.2x |
| > 1000s | 59 | 3.9x |

- Gurobi V3.0 versus V2.0 (P=4) (bigger test set)

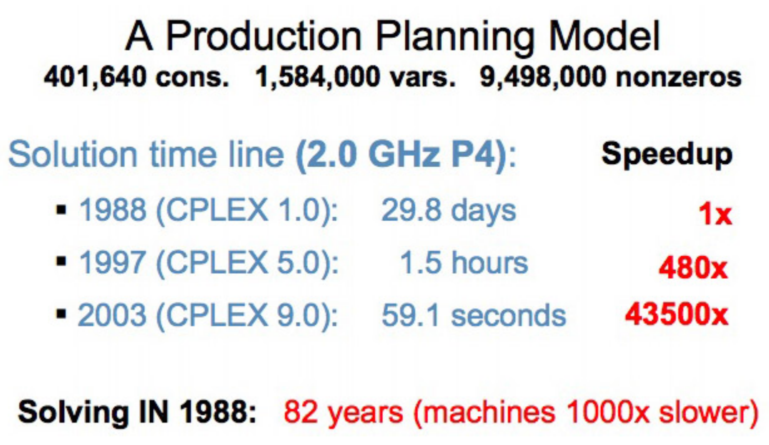
| Time | # Models | Speedup |
|---|---|---|
| > 1s | 794 | 1.6x |
| > 10s | 521 | 2.0x |
| > 100s | 295 | 2.9x |
| > 1000s | 144 | 6.7x |

Figure 34: Speedups of Gurobi MIP solver between versions on problems of several sizes. Reprinted from Bixby (2010).

### 8.1.3. Nonspecific Reports of Large Speedups

Martin Grötschel points to one production-planning problem which he claims became on average around twice as fast to solve every year between 1988 and 2003. Figure 35 is taken from his (2009, p. 31) talk about this. His numbers translate to an average 2-times speedup per year for 1988–1997, and 2.1-times speedup per year for 1997–2003. This example was very likely chosen to exhibit large speedups.

Bixby (2010) cites 3,300-times speedups (figure 36) in a type of linear programming (seemingly) over a non-specified time period before 2004, then little between 2004 and 2011. In the same earlier time, hardware improved times by a factor of 1,600, which gives us an idea of the time frame. One could investigate further using the journal information.



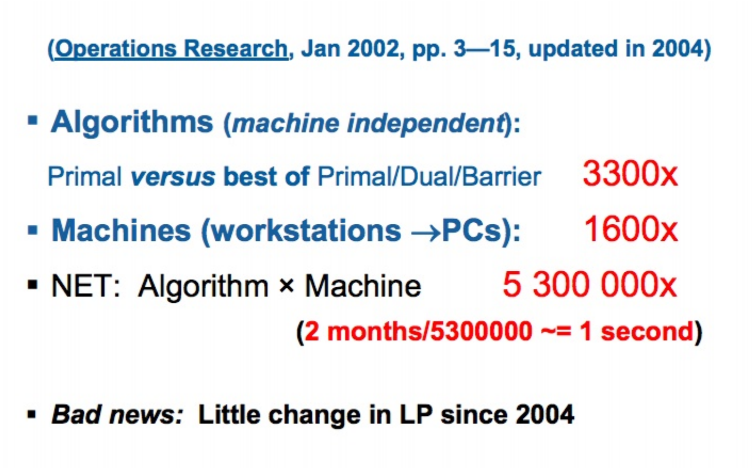
A Production Planning Model

**401,640 cons. 1,584,000 vars. 9,498,000 nonzeros**

| Solution time line **(2.0 GHz P4)**: | | **Speedup** |
|---|---|---|
| ▪ 1988 (CPLEX 1.0): | 29.8 days | **1x** |
| ▪ 1997 (CPLEX 5.0): | 1.5 hours | **480x** |
| ▪ 2003 (CPLEX 9.0): | 59.1 seconds | **43500x** |

**Solving IN 1988:** 82 years (machines 1000x slower)

Figure 35: Martin Grötschel's account of speedup in one production-planning problem. Reprinted from Grötschel (2009).



**(Operations Research, Jan 2002, pp. 3—15, updated in 2004)**

- **Algorithms (*machine independent*):**
  Primal ***versus*** **best of** Primal/Dual/Barrier 3300x
- **Machines (workstations →PCs):** 1600x
- NET: Algorithm × Machine 5 300 000x
  **(2 months/5300000 ~= 1 second)**

- ***Bad news:*** **Little change in LP since 2004**

Figure 36: Algorithmic improvement relative to hardware improvement for an unspecified class of linear programming problems. Reprinted from Bixby (2010).

## 8.2. Scheduling

### 8.2.1. University of Nottingham Employee Scheduling Benchmarks

The University of Nottingham maintains a dataset of performance of published algorithms on instances from their own benchmark set, as reported in the papers where the algorithms were published.

Below is the collected data. Figure 37 shows the best scores so far over time on each of a set of benchmarks, divided by the worst score recorded in the dataset for that benchmark. Dividing by the worst score is intended to make the scores more comparable, as different benchmarks have different possible ranges of scores. A higher score is worse, all things equal. However, note that authors often had goals other than minimizing the score—for example, for their algorithm to work on a broad range of problems.

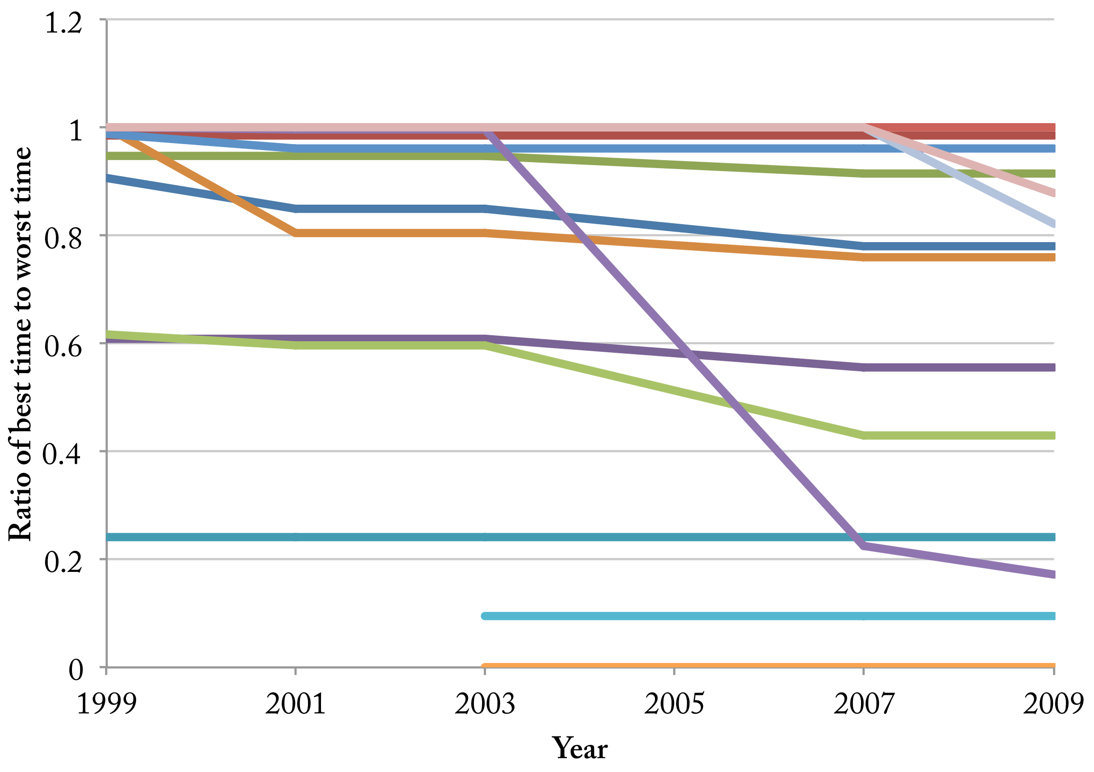


Figure 37: Scores achieved relative to worst recorded scores on the Employee Scheduling Benchmarks. Each line represents a different algorithm. Data from ASAP (2013).

Hardware was not kept constant, and generally was the standard hardware available at the time.

The original dataset has forty-eight benchmarks and sixteen algorithms, published over nine years; however, most of these combinations are missing. Benchmarks with fewer than two data points are excluded, as are benchmarks which were already completed perfectly the earliest time they were measured, and benchmarks for which all the data was for a single year. Note that the figure shows the best performance so far—often there are cases of worse performance in later years, which are not visible in this format. Also omitted is a single 1984 record of performance on the Musa benchmark. Its value is used as the worst performance in the graph below; the line just doesn't run back to 1984.

Note that flat progress generally means there were only two data points, and the second one was worse than the first. This is arguably not much more evidence of slow progress on a benchmark than only having a single data point, since algorithms that will solve a given problem slowly always exist. Such benchmarks are included here, however, while those with only one data point are not, because it suggests some effort was being made on that benchmark, and that progress was not being made; where there is only one data point, progress may just not have been reported.

## 9. Machine Learning

### 9.1. General Impressions from the Field

Computer scientist Ernest Davis's impression is that diminishing returns are a common trend in both natural language understanding and computer vision:

> For most tasks in automated vision and natural language processing, even quite narrowly defined tasks, the quality of the best software tends to plateau out at a level considerably below human abilities, though there are important exceptions. Once such a plateau has been reached, getting further improvements to quality is generally extremely difficult and extremely slow. (Davis 2012)

Roger K. Moore (2012) has investigated progress in transcription, and also claims it is slowing. He says it appears to be approaching an asymptote with error rates of 20–50% in conversational speech.

Davis expands upon the state of machine learning in natural language and vision in an email message to Chris Hallquist:

> A. Certainly startling improvements do sometimes occur. A notable recent example was Geoff Hinton's success at Pascal VOC which, I understand, blew previous work and the competition out of the water.
>
> B. My impression is mostly drawn from work in NLP and information extraction. I see a fair amount of this, though I don't follow it in technical detail, and certainly don't systematically keep track of numbers. My impression is that a common pattern here is:
>
> 1. There is an almost trivial algorithm that sets a medium high baseline. For instance, pronoun resolution to the most recent possible antecedent works something like 75% or 80% of the time. Lexical ambiguity resolution to the most frequent meaning succeeds some significant fraction of the time. The early (late '50s, early '60s) machine translation efforts got promising results on toy corpora.
> 2. There are some low-hanging fruit that with a medium amount of work will give you a notable improvement on (1). For instance, n-gram analysis and some simple selectional restrictions will give you some significant improvement on resolving lexical ambiguity.
> 3. With very substantial effort, tons of data, tons of statistical analysis, and state-of-the art machine learning techniques, you can push (2) up some-

what. For instance, the work I've seen recently on machine translation has involved using huge bitexts, extracting every pattern you can identify, and then tuning like crazy using ML techniques. I don't know to what extent that applies to Google Translate, though.

4. And then things get really tough. All you can do in the current state of things is push on (3), and my sense is you get steadily diminishing returns. For example 3 or 4 years ago, we had some interviews by researchers in machine translation, all of whom were pursuing some form or another of (3). One of them had gotten a BLEU score of .37 where the competition had gotten .35—that was the order of number, though I don't remember details. It didn't seem very impressive to me, but my colleague Ralph Grishman, who is the expert in our department, said that getting 2 points on the BLEU score is really hard.

   Another specific example: Jerry DeJong's FRUMP (1979) was the first information extraction [IE] system (before the term was coined). It achieved something like 40% accuracy on the facts extracted from newspaper stories about earthquakes and some other categories. Current IE systems 35 years later achieve accuracies in the mid 60's I think. Now, these are quite different and incomparable numbers; I am not claiming that FRUMP would score 40% under current NIST test conditions. It was a proof-of-concept (i.e. toy) program. On the other hand, it was written by a single graduate student as a doctoral thesis, and used no statistical or ML techniques. And my impression is that improvement here among state-of-the-art programs has been extremely slow. IE has been a matter of very extensive study for at least the last 20 years; it is a highly circumscribed form of the general natural language understanding problem; and here we are still at 60%-ish depending on what task you're talking about.

   In answer to ["Does this apply mainly to the 'hard for computers, easy for humans' category of problems, or are there many cases of it applying to other kinds of problems?"], I can't think of any very convincing cases in CS where this comes up outside AI. A couple of possible candidates, perhaps worth some thought. [Mentions programming languages, software engineering, educational software.]

### 9.2. Transcription

#### 9.2.1. Transcription NIST Automatic Speech Recognition Evaluations

Figure 38 shows a historical summary from the NIST Information Technology Laboratory Information Access Division.

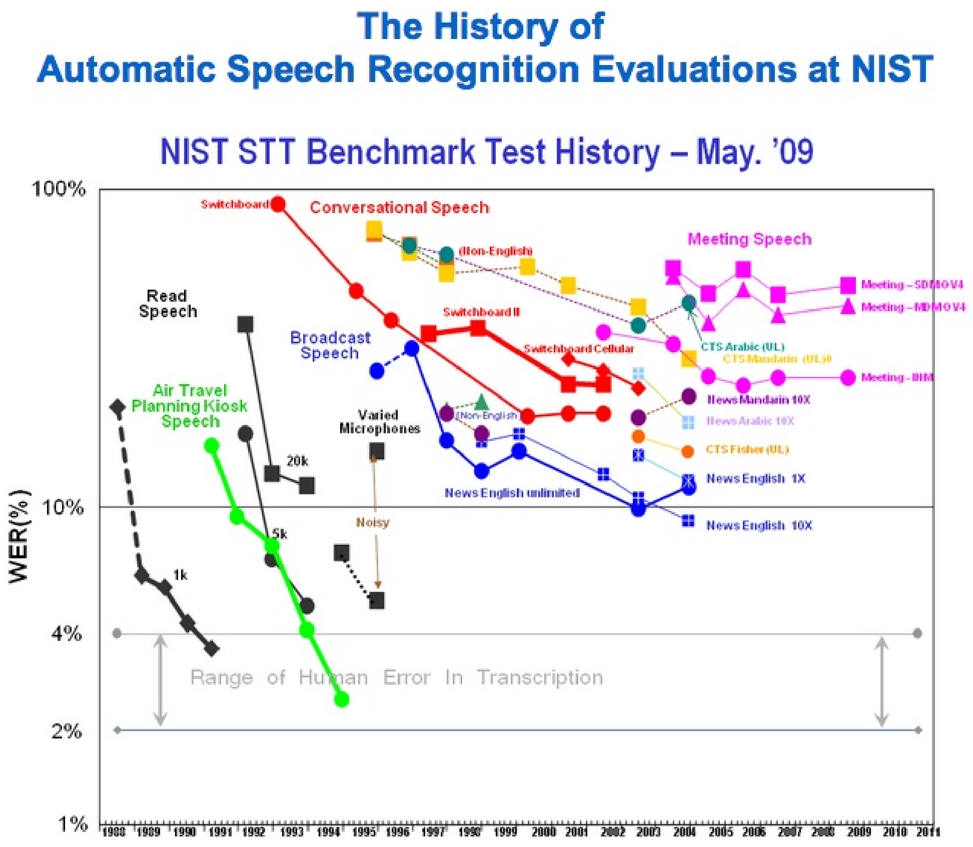


Figure 38: NIST speech recognition testing results for a variety of transcription tasks. Reprinted from NIST (2009b).

#### 9.2.2. Training Data

In two studies, word error rates decreased linearly with exponential training data (Lamel, Gauvain, and Adda 2000, 2002). The task was transcription of continuous broadcast news by state-of-the-art, large-vocabulary continuous speech recognition (LVCSR) systems. Extrapolations from a paper by Roger K. Moore (2003) suggest zero errors should be attained in that task after between 600,000 and 10,000,000 hours of training data. Figures 39 and 40 below show the results and extrapolations.

### 9.3. Parsing

Computerized parsing algorithms were first designed in the '50s (Russell and Norvig 2010, p. 920), but empirical evaluations on real text prior to 1989 are not available.

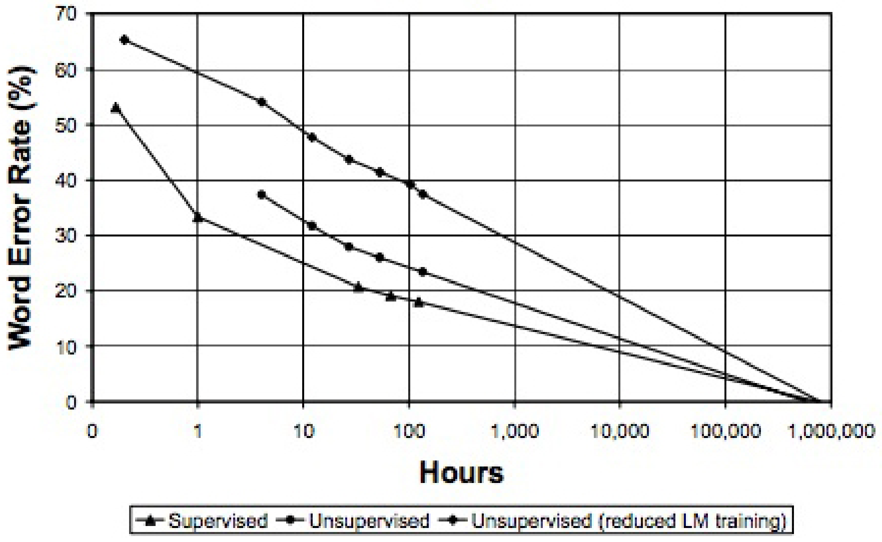


Figure 39: Word error rates by quantity of training data for a transcription task, using three methods, extrapolated to perfection. Reprinted from Moore (2003).

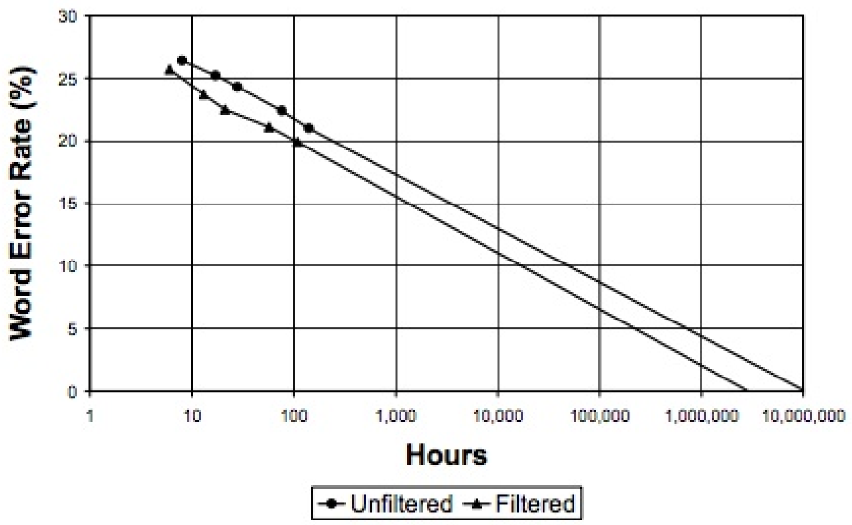


Figure 40: Word error rates by quantity of training data for a transcription task, using two methods, extrapolated to perfection. Reprinted from Moore (2003).

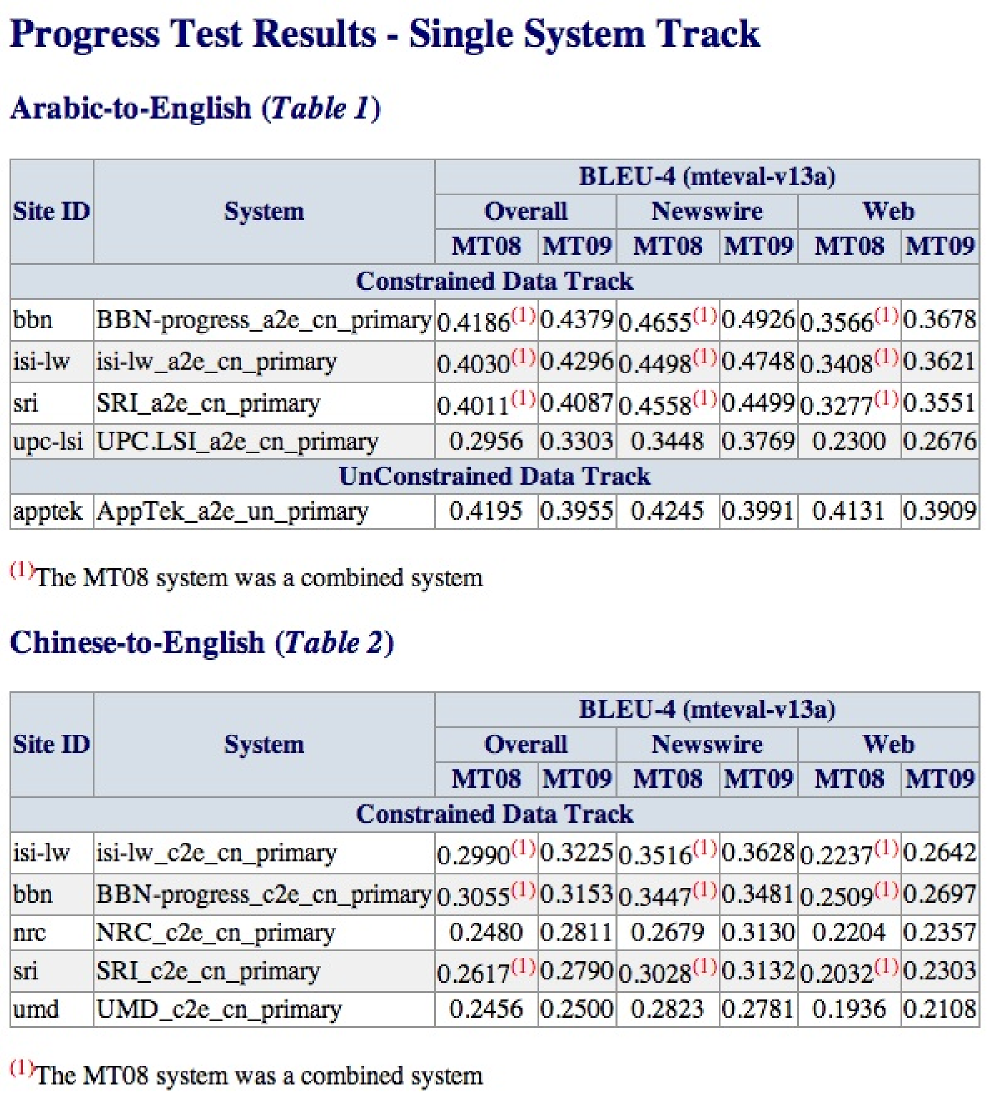
A rough history of reported parsing accuracies:

- 1950s: first computerized parsing algorithms (Russell and Norvig 2010, p. 920)
- 1989: **60–77%** accuracy (Salton and Smith 1989)
- 1992: **90%** (Magerman and Weir 1992)
- 2005: **92.0%** (Russell and Norvig 2010, p. 920)
- 2007: **90.6%** (Russell and Norvig 2010, p. 920)
- 2008: **93.2%** (Russell and Norvig 2010, p. 920)

Regarding the last three, Russell and Norvig (2010) warn: “These numbers are not directly comparable, and there is some criticism of the field that it is focusing too narrowly on a few select corpora, and perhaps overfitting on them.”

## 9.4. Translation

Figure 41 is taken from the NIST 2009 Open Machine Translation Evaluation. It compares BLEU scores across years from sites that participated in both of the 2008 and 2009 NIST open machine translation evaluations, in either Arabic to English or Chinese to English. Most of the sites scored better in 2009.



**Progress Test Results - Single System Track**

**Arabic-to-English (*Table 1*)**

| Site ID | System | BLEU-4 (mteval-v13a) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Overall | | Newswire | | Web | |
| | | MT08 | MT09 | MT08 | MT09 | MT08 | MT09 |
| **Constrained Data Track** | | | | | | | |
| bbn | BBN-progress_a2e_cn_primary | 0.4186(1) | 0.4379 | 0.4655(1) | 0.4926 | 0.3566(1) | 0.3678 |
| isi-lw | isi-lw_a2e_cn_primary | 0.4030(1) | 0.4296 | 0.4498(1) | 0.4748 | 0.3408(1) | 0.3621 |
| sri | SRI_a2e_cn_primary | 0.4011(1) | 0.4087 | 0.4558(1) | 0.4499 | 0.3277(1) | 0.3551 |
| upc-lsi | UPC.LSI_a2e_cn_primary | 0.2956 | 0.3303 | 0.3448 | 0.3769 | 0.2300 | 0.2676 |
| **UnConstrained Data Track** | | | | | | | |
| apptek | AppTek_a2e_un_primary | 0.4195 | 0.3955 | 0.4245 | 0.3991 | 0.4131 | 0.3909 |

(1)The MT08 system was a combined system

**Chinese-to-English (*Table 2*)**

| Site ID | System | BLEU-4 (mteval-v13a) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Overall | | Newswire | | Web | |
| | | MT08 | MT09 | MT08 | MT09 | MT08 | MT09 |
| **Constrained Data Track** | | | | | | | |
| isi-lw | isi-lw_c2e_cn_primary | 0.2990(1) | 0.3225 | 0.3516(1) | 0.3628 | 0.2237(1) | 0.2642 |
| bbn | BBN-progress_c2e_cn_primary | 0.3055(1) | 0.3153 | 0.3447(1) | 0.3481 | 0.2509(1) | 0.2697 |
| nrc | NRC_c2e_cn_primary | 0.2480 | 0.2811 | 0.2679 | 0.3130 | 0.2204 | 0.2357 |
| sri | SRI_c2e_cn_primary | 0.2617(1) | 0.2790 | 0.3028(1) | 0.3132 | 0.2032(1) | 0.2303 |
| umd | UMD_c2e_cn_primary | 0.2456 | 0.2500 | 0.2823 | 0.2781 | 0.1936 | 0.2108 |

(1)The MT08 system was a combined system

Figure 41: Across-year comparison of participants in the 2008 and 2009 NIST open machine translation evaluations, for Arabic to English and Chinese to English. Reprinted from NIST (2009a).

## 9.5. Vision

### 9.5.1. Facial Recognition

Figure 42 shows the reduction in error rate for state-of-the-art face recognition over the years according to the FERET, FRVT 2002, and FRVT 2006 evaluations (Grother, Quinn, and Phillips 2011). Note that it documents the false nonmatch rate when the false match rate is 0.001; for a higher false match rate, the false nonmatch rate should be lower.

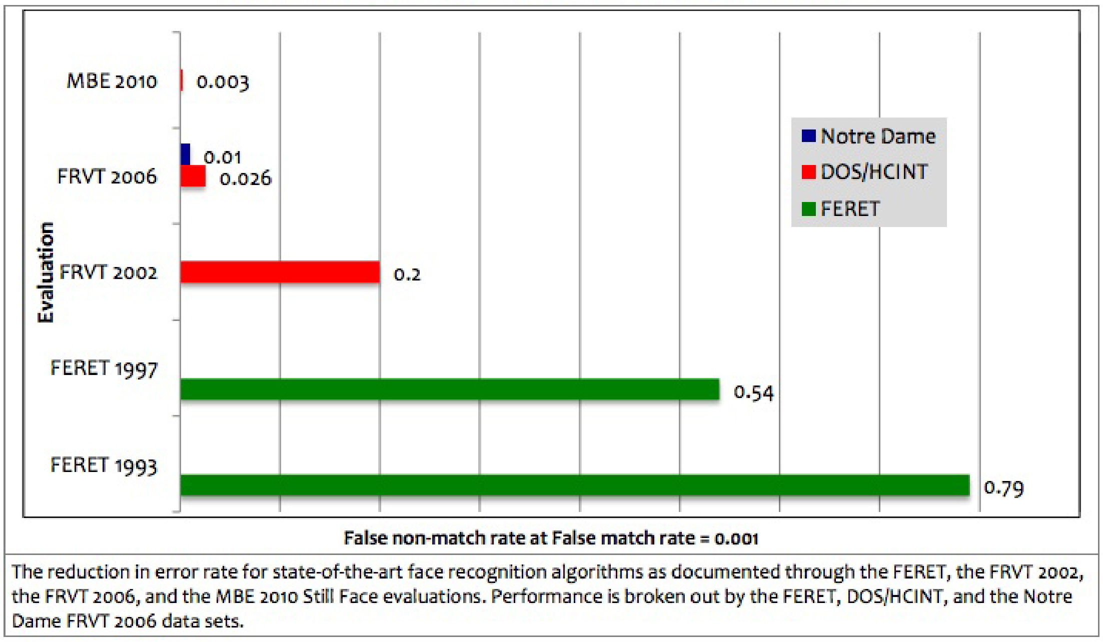


Figure 42: False nonmatch rate when the false match rate is 0.001 for state-of-the-art facial recognition algorithms on three datasets. Data from Grother, Quinn, and Phillips (2011, fig. 28).

Efforts in this area have most likely increased in the last decade, as US government support for biometrics has grown. Between the fiscal years 2007 and 2011, for instance, annual US Department of Defense funding for biometrics grew every year, from $376.8 million to $769.7 million (GAO 2009).

### 9.5.2. Object and Scene Recognition

Object and scene recognition can be divided into several types of task. *Classification* involves judging whether there is an object of some type in a given picture. *Detection* involves determining where the object is in a given picture. *Segmentation* means simplifying an image into parts, usually according to useful objects or boundaries.

## Classification

PASCAL Visual Object Classes (VOC) challenges began in 2004. In some years, participating algorithms were run on datasets from previous years to allow comparison. Figure 43 shows progress on classification since 2009 on the 2009 dataset. Figure 44 shows maximum average precision over time on the 2008 dataset.

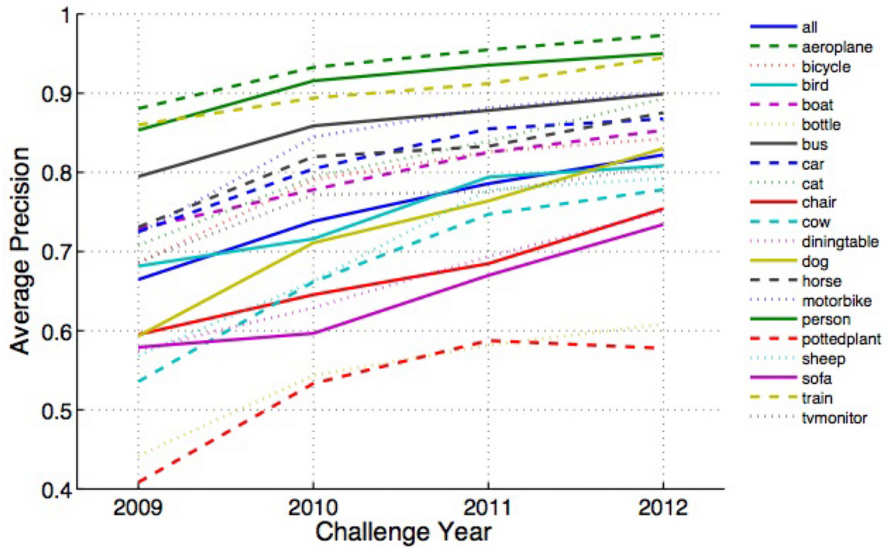


Figure 43: Progress in precision on classification tasks in PASCAL VOC challenges. Reprinted from Williams (2012).

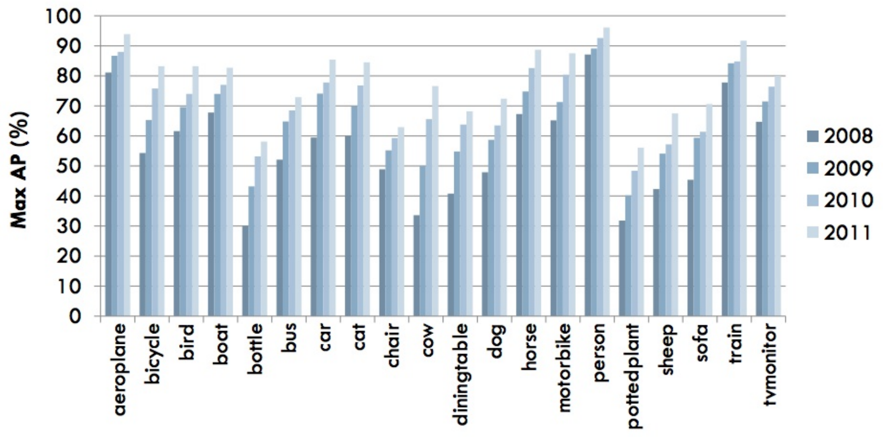


Figure 44: Progress in maximum average precision for classification of various objects in PASCAL VOC challenges. Reprinted from Everingham (2011a).

## Detection

Figures 45 and 46 are analogous to the previous two figures. Progress was slow in 2011; negative progress is highlighted.

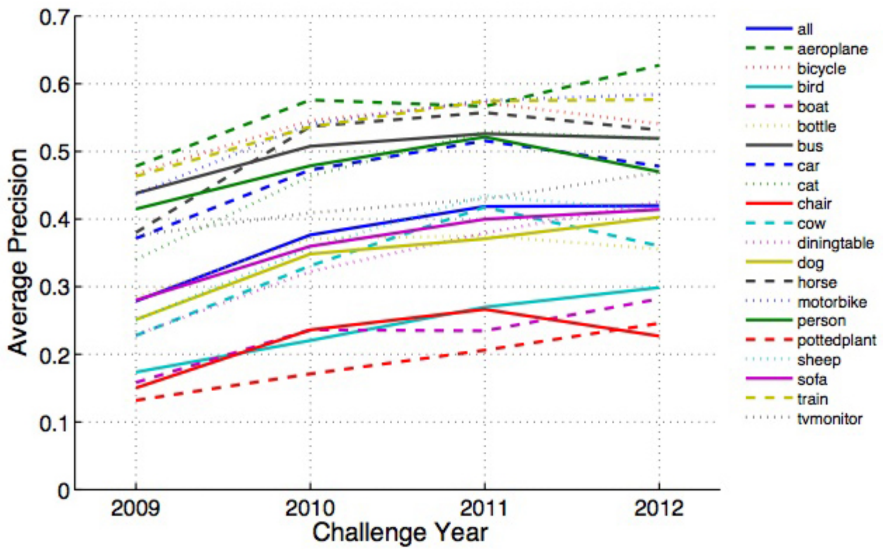


Figure 45: Progress in precision on detection tasks in PASCAL VOC challenges. The 2009 dataset is used in all years. Reprinted from Williams (2012).

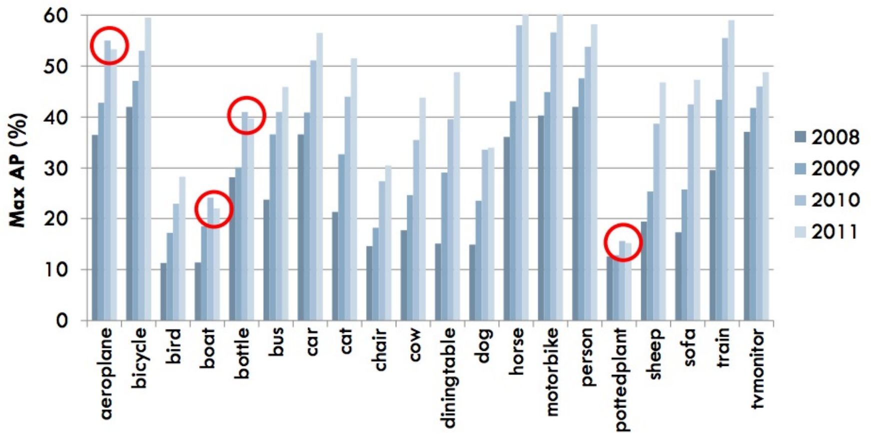


Figure 46: Maximum average detection precision on a variety of objects over time. Reprinted from Everingham (2011b).

## Other Tasks

Classification and detection are central parts of the PASCAL VOC challenge. Figures 47 and 48 show progress in some peripheral tasks included in some years.

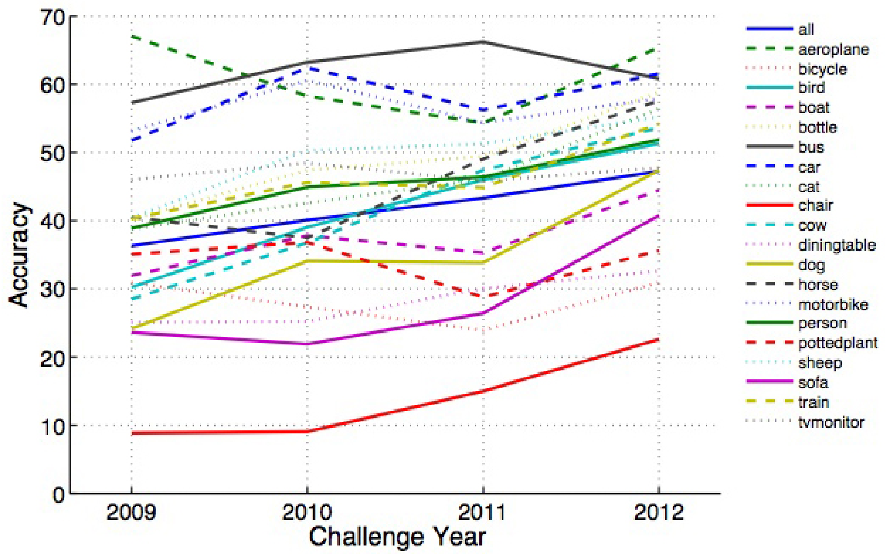


Figure 47: Progress in segmentation accuracy percentage in PASCAL VOC challenges. All tests use their 2009 dataset. Reprinted from Williams (2012).

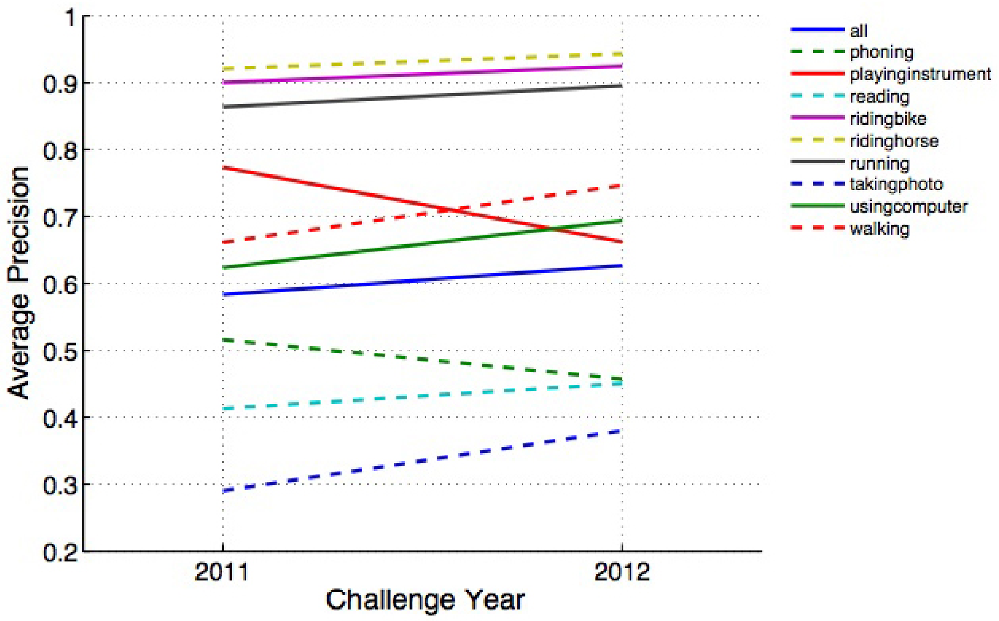


Figure 48: Progress in action classification in PASCAL VOC challenges. All tests use their 2011 dataset. Reprinted from Williams (2012).

## Training Data

Figures 49 and 50, taken from slides by Ramanan (2012), show some effects of more training data on precision in object recognition.

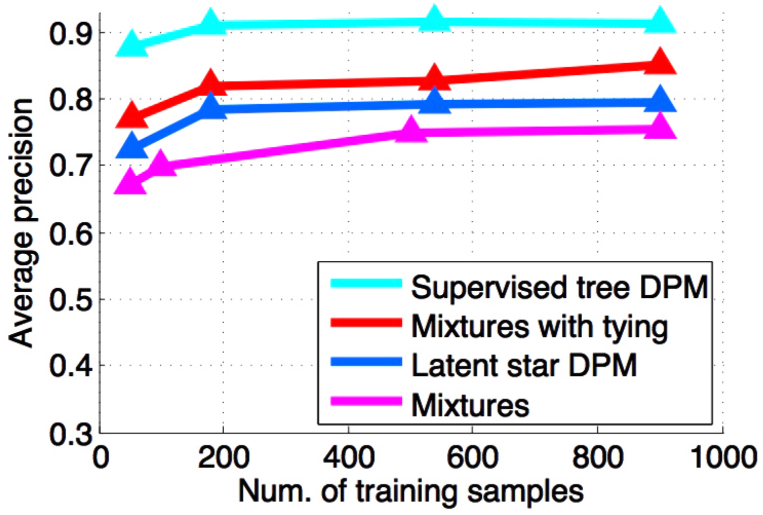


Figure 49: Effect of training samples on face recognition precision for various methods. Reprinted from Ramanan (2012).

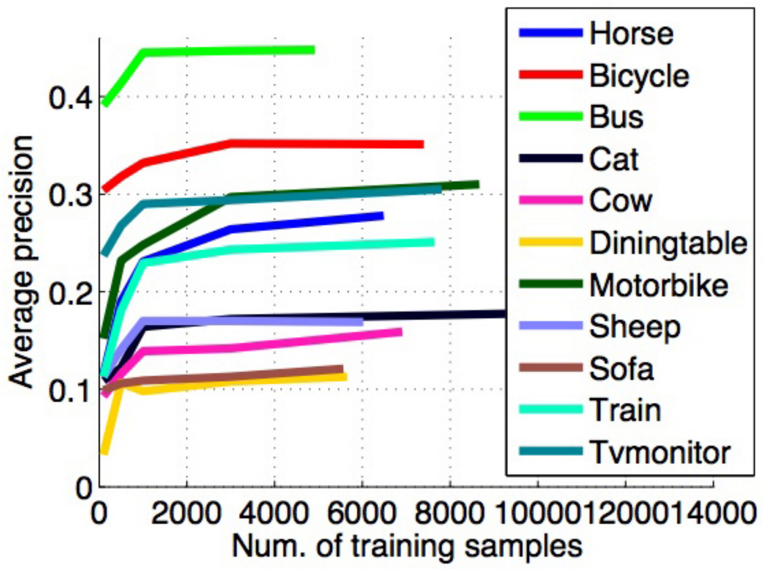


Figure 50: Effect of training samples on object recognition precision for various methods. Reprinted from Ramanan (2012).

## Anecdotal Evidence on Jumps in Vision Progress

Le et al. (2012) claim a "15.8% accuracy for object recognition on ImageNet with 20,000 categories, a significant leap of 70% relative improvement over the state-of-the-art." In a 2013 email to Chris Hallquist, Yann LeCun says:

> The Google cat detector thing is totally passé. The recent defining moment in computer vision is the smashing victory by Geoff Hinton and his team on the ImageNet Large Scale Visual Recognition Challenge last October. The entire CV community was stunned that they could get 15% error when everyone else gets 25%. Google was stunned by the result and bought the company Geoff formed with his students (what they bought was Geoff and his students). The method they use is my good old convolutional net trained with backprop (and a few tricks). What made their success is a ridiculously efficient implementation on GPUs, which allowed them to train a model in about a week on a single machine, and experiment with various tricks and settings. That's clearly a success of "big data" in the sense that they use a relatively well established method whose complexity scales sub-linearly with the data size, sped up the code, trained on a large dataset, and spent 8 months trying out various tricks to improve the results.

## 10. Conclusions and Future Work

We have looked at progress in six areas of algorithms research. Many of these areas have seen fast improvement, with algorithmic progress contributing 50–100% as much as hardware progress. SAT solvers take in the realm of ten percent less time per year on average. Mixed integer programming algorithms investigated here doubled in speed every year. Chess tends to get better by around fifty Elo points per year, and Go by a single stone per year. These point to algorithmic progress and hardware progress contributing roughly equally. The physics simulations reported on are purported to see roughly equal improvement from hardware and algorithms. Factoring has made substantial progress, but it is unclear how much of this is accounted for by hardware. In many areas, machine learning is making steeply diminishing progress in terms of percentage accuracy.

Gains in particular years can vary substantially, though are rarely zero in any of these areas. On the scale of a few years, chess, Go, and factoring seem to make fairly consistent amounts of progress. Progress on different SAT problems varies widely between taking almost as much time two years later, and taking a tiny fraction of the time. The current data does not say a lot about long-term trajectories for particular problems, though—it could be that some consistently see a lot of improvement, or that all of them occasionally do. However, there seems to be a persistent trend of problems that initially take longer seeing more progress than those that are initially faster. The small amount of data on MIP progress says some software versions see more than five times as much progress as other versions. The jumpiness of progress in physics simulations is ambiguous from the present data.

This has been a preliminary survey. There are a number of ways to proceed. One is to investigate other areas of algorithms in a similar manner—for instance, combinatorial optimization, signal processing, or Scrabble. However, it may be better to get a surer picture of progress on the algorithms here before venturing more tentative estimates. There are many apparently promising sources of data on the current areas of algorithms not investigated for this report.

This report investigated data from the SAT competition over three competitions, but they hold data for seven. Analyzing this additional data would give a much clearer picture of progress there. In general, ongoing competitions and challenges are valuable sources of data.

For many of these areas there are papers comparing small numbers of algorithms, published at different times. For instance, there is apparently a string of papers in SAT conferences by Holger Hoos and his students and Chu Min Li's group comparing state-of-the-art solvers at different times, which may offer a somewhat independent check or longer history.

Another approach to many of these areas is to acquire historical algorithms and run them oneself on modern hardware. This could give more reliable data and also allow hardware and software contributions to be identified.

For each of these algorithms, it would be useful to know more about the inputs that produced the current progress, and about the social interactions surrounding them. This may be difficult to determine accurately, but, for instance, sociological work on chess AI researchers seems to exist, so a basic picture may be relatively cheap to acquire.

The current findings have interesting implications for AI trajectories. At this margin there is much opportunity for further work and research.

## Acknowledgments

Many thanks to Paul Christiano and Luke Muehlhauser for helpful ideas, conversation, and feedback. Thanks also to Chris Hallquist and Charmin Sia for collecting and collating information, and to Benjamin Noble, Alex Vermeer, and Caleb Bell for help editing and preparing this document. I am also grateful to many people for directing me to useful information and helping me to interpret it; thanks for this to Nick Wedd, Petr Baudis, Ed Lazowska, David Fotland, Brian Borchers, Hideki Kato, Daniel Le Berre, Olivier Roussel, Tim Curtois, Martin Grötschel, Jesse Liptrap, João Marques-Silva, Bart Selman, Holger Hoos, Hans Mittelman, Patrick Grace, Armin Biere, Niklas Een, Robert Hyatt, Jonathan Chin, Russ Greiner, David Keyes, Jason Eisner, Carl Shulman, and Kaj Sotala.

## References

Automated Scheduling, Optimisation and Planning Group. 2013. "Employee Scheduling Benchmark Data Sets." University of Nottingham. Accessed July 22, 2013. `http://www.cs.nott.ac.uk/~tec/NRP/`.

Banks, Graham. 2013. "Re: Questions for a Research Project." CCRL Discussion Board. April 22. Accessed July 22, 2013. `http://kirill-kryukov.com/chess/discussion-board/viewtopic.php?f=7&t=6999`.

Baudiš, Petr, and Jean-loup Gailly. 2011. "PACHI: State of the Art Open Source Go Program." In *Advances in Computer Games: 13th International Conference, ACG 2011,* edited by H. Jaap van den Herik and Aske Plaat, 24–38. Lecture Notes in Computer Science 7168. Tilburg, The Netherlands: Springer, November 20–22. doi:`10.1007/978-3-642-31866-5_3`.

Bixby, Robert E. 2010. "Latest Advances in Mixed-Integer Programming Solvers." Lecture given at the Spring School on Combinatorial Optimization in Logistics, Université de Montréal, May 19. `http://symposia.cirrelt.ca/system/documents/0000/0136/Bixby.pdf`.

Chess Engines Grand Tournament. 2006–2012. "CEGT 40/4 (2Ghz): All Versions – Min. 30 Games." Archived versions. As revised July 1, 2006–March 1, 2012. `http://web.archive.org/web/*/http://www.husvankempen.de/nunn/40_4_Ratinglist/40_4_AllVersion/rangliste.html`.

ChessBase. 2013. "Houdini 3: The World's Strongest Chess Engine in the Fritz Interface." Interview with Robert Houdart, author of Houdini, September 29. `http://en.chessbase.com/home/TabId/211/PostId/4008591`.

Coles, L. Stephen. 2002. "Computer Chess: The Drosophila of AI." *Dr. Dobb's Journal,* October 30. `http://www.drdobbs.com/parallel/computer-chess-the-drosophila-of-ai/184405171`.

Computer Chess Rating Lists Team. 2006–2013. "CCRL 40/40: Complete List." Archived versions. As revised May 31, 2006–January 13, 2013. `http://web.archive.org/web/*/http://computerchess.org.uk/ccrl/4040/rating_list_all.html`.

Computer Chess Rating Lists Team. 2013. "40/40 Downloads and Statistics: Rybka." Accessed July 22, 2013. http://www.computerchess.org.uk/ccrl/4040/cgi/compare_engines.cgi?family=Rybka&print=Rating+list&print=Results+table&print=LOS+table&print=Ponder+hit+table&print=Eval+difference+table&print=Comopp+gamenum+table&print=Overlap+table&print=Score+with+common+opponents.

Contini, Scott. 2010. "General Purpose Factoring Records." FactorWorld. Accessed July 22, 2013. http://www.crypto-world.com/FactorRecords.html.

Dailey, Don. 2008. "9x9 Scalability study." Computer Go Server. February 13. Accessed July 22, 2013. http://cgos.boardspace.net/study/.

Davis, Ernest. 2012. "The Singularity and the State of the Art in Artificial Intelligence." Working Paper, New York, May 9. Accessed July 22, 2013. http://www.cs.nyu.edu/~davise/papers/singularity.pdf.

Everingham, Mark. 2011a. "Overview and Results of the Classification Challenge." Paper presented at the PASCAL Visual Object Classes Challenge Workshop, ICCV 2011, Barcelona, Spain, November 7. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2011/workshop/voc_cls.pdf.

———. 2011b. "Overview and Results of the Detection Challenge." Paper presented at the PASCAL Visual Object Classes Challenge Workshop, ICCV 2011, Barcelona, Spain, November 7. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2011/workshop/voc_det.pdf.

Fotland, David. 2002. "David Fotland." Smart Games. February 14. Accessed July 21, 2013. http://www.smart-games.com/david.html.

GAO (Government Accountability Office). 2009. *Defense Biometrics: DOD Can Better Conform to Standards and Share Biometric Information with Federal Agencies.* Washington, DC, March 31. http://www.gao.gov/products/GAO-11-276.

Grother, Patrick J., George W. Quinn, and P J. Phillips. 2011. *Report on the Evaluation of 2D Still-Image Face Recognition Algorithms.* NIST Interagency Report 7709. National Institute of Standards and Technology, August 24. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=905968.

Grötschel, Martin. 2009. "Combinatorial Optimization in Action." Lecture given at the Winter 2009 MathAcrossCampus Colloquium, University of Washington, Seattle, WA, January 22. http://www.math.washington.edu/mac/talks/20090122SeattleCOinAction.pdf.

Houdart, Robert. 2012. "Houdini Chess Engine." Accessed July 22, 2013. http://www.cruxis.com/chess/houdini.htm.

Karlsson, Thoralf. 2008. "Comments to the Swedish Rating List 1." Swedish Chess Computer Association. Archived version. As revised September 26. http://web.archive.org/web/20081016112102/http://ssdf.bosjo.net/comment.htm.

Keyes, David, Robert Armstrong, David Bailey, John Bell, E. Wes Bethel, David Brown, Phillip Colella, et al. 2004. *A Science-Based Case for Large-Scale Simulation, Volume 2.* New York: US Department of Energy's Office of Science, September 14. http://science.energy.gov/~/media/ascr/pdf/program-documents/archive/Scales_report_vol2.pdf.

KGS Go Server. 2010. "KGS Game Archives: Games of KGS player zen19." Archived version. As revised December 3. http://web.archive.org/web/20101203045501/http://www.gokgs.com/graphPage.jsp?user=Zen19.

———. 2013a. "KGS Game Archives: Games of KGS player zen19." Accessed July 22, 2013. `http://www.gokgs.com/gameArchives.jsp?user=zen19d`.

———. 2013b. "Tournament." Accessed July 22, 2013. `http://www.gokgs.com/help/tournaments.html`.

Lamel, Lori, Jean-Luc Gauvain, and Gilles Adda. 2000. "Lightly Supervised Acoustic Model Training." In *Proceedings of the ISCA ITRW ASR2000: Automatic Speech Recognition: Challenges for the New Millenium,* edited by Martine Adda-Decker, 150–154. Paris, France: Nouvelle Imprimerie Laballery, September 18–20. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.2392&rep=rep1&type=pdf`.

———. 2002. "Lightly Supervised and Unsupervised Acoustic Model Training." *Computer Speech & Language* 16 (1): 115–129. doi:`10.1006/csla.2001.0186`.

Le, Quoc V., Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. 2012. "Building High-level Features Using Large Scale Unsupervised Learning." In *Proceedings of the 29th International Conference on Machine Learning (ICML-12),* edited by John Langford and Joelle Pineau, 81–88. Edinburgh, Scotland: Omnipress, June 27–July 3. `http://arxiv.org/pdf/1112.6209v5.pdf`.

Levy, David, and Monty Newborn. 1991. *How Computers Play Chess.* 1st ed. New York: Computer Science Press.

Magerman, David M., and Carl Weir. 1992. "Efficiency, Robustness and Accuracy in Picky Chart Parsing." In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics (ACL '92),* edited by Henry S. Thompson, 40–47. Newark, DE: Association for Computational Linguistics, June 28–July 2. doi:`10.3115/981967.981973`.

Marques-Silva, João. 2010. "Boolean Satisfiability Solving: Past, Present & Future." Presentation given at the Microsoft Research International Workshop on Tractability, Cambridge, UK, July 5–6. `http://research.microsoft.com/en-us/events/tractability2010/joao-marques-silva-tractability2010.pdf`.

Moore, Roger K. 2003. "A Comparison of the Data Requirements of Automatic Speech Recognition Systems and Human Listeners." In *Eurospeech 2003 Proceedings: 8th European Conference on Speech Communication and Technology,* 2581–2584. Geneva, Switzerland: ISCA, September 1–4. `http://www.dcs.shef.ac.uk/~roger/publications/Moore%20-%20Spoken%20language%20interaction%20with%20intelligent%20systems.pdf`.

———. 2012. "Spoken Language Interaction with 'Intelligent' Systems: How Are We Doing, and What Do We Need to Do Next?" Presentation given at the EUCogII Workshop on Challenges for Artificial Cognitive Systems II, Oxford, January 21. `http://www.dcs.shef.ac.uk/~roger/publications/Moore%20-%20Spoken%20language%20interaction%20with%20intelligent%20systems.pdf`.

Muehlhauser, Luke. 2011. "Historical Chess Engines' Estimated ELO Ratings." Last revised October 9. `http://lukeprog.com/special/chess.pdf`.

Muehlhauser, Luke, and Anna Salamon. 2012. "Intelligence Explosion: Evidence and Import." In *Singularity Hypotheses: A Scientific and Philosophical Assessment,* edited by Amnon Eden, Johnny Søraker, James H. Moor, and Eric Steinhart. The Frontiers Collection. Berlin: Springer.

National Institute of Standards and Technology Multimodal Information Group. 2009a. "2009 Open Machine Translation Evaluation: Official Release of Results." October 27. Accessed July 22, 2013. `http://www.itl.nist.gov/iad/mig/tests/mt/2009/ResultsRelease/progress.html`.

National Institute of Standards and Technology Multimodal Information Group. 2009b. "The History of Automatic Speech Recognition Evaluations at NIST." September 14. Accessed July 22, 2013. `http://www.itl.nist.gov/iad/mig/publications/ASRhistory/index.html`.

Pomerance, Carl. 1996. "A Tale of Two Sieves." *Notices of the American Mathematical Society,* December, 1473–1485. `http://www.ams.org/notices/199612/pomerance.pdf`.

Ramanan, Deva. 2012. "Deformable Part Models: Some Thoughts on Why They Work and What Next." Paper presented at the PASCAL Visual Object Classes Challenge Workshop, ECCV 2012, Florence, Italy, October 12. `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/workshop/deva.ramanan.pdf`.

Russell, Stuart J., and Peter Norvig. 2010. *Artificial Intelligence: A Modern Approach.* 3rd ed. Upper Saddle River, NJ: Prentice-Hall.

Salton, G., and M. Smith. 1989. "On the Application of Syntactic Methodologies in Automatic Text Analysis." In *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 137–150. Cambridge, MA.

SAT Competition Organizers. 2007–2011. "The International SAT Competitions Web Page." Accessed July 22, 2013. `http://www.satcompetition.org/`.

*Sensei's Library.* 2008, s.v. "Ing Prize." As revised September 19. `http://senseis.xmp.net/?IngPrize:v23`.

———. 2013, s.v. "Computer Go." As revised May 26. `http://senseis.xmp.net/?ComputerGo:v142`.

———. 2013, s.v. "KGSBotRatings." As revised January 7. `http://senseis.xmp.net/?KGSBotRatings:v182`.

Sonas, Jeff. 2003. "Man vs Machine: Who is Winning?" *ChessBase,* October 8. `http://en.chessbase.com/home/TabId/211/PostId/4001229`.

Swedish Chess Computer Association. 2007–2012. "The SSDF Rating List." Archived versions. As revised August 11, 2007–September 9, 2012. `http://web.archive.org/web/*/http://ssdf.bosjo.net/list.htm`.

Wedd, Nick. 2013. "Human-Computer Go Challenges." Accessed July 22, 2013. `http://www.computer-go.info/h-c/`.

*Wikipedia.* 2013, s.v. "Swedish Chess Computer Association." Accessed May 12. `http://en.wikipedia.org/w/index.php?title=Swedish_Chess_Computer_Association&oldid=554696080`.

Williams, Chris. 2012. "Introduction: History and Analysis." Introduction to Part II of VOC 2005-2012: The VOC Years and Legacy. At the PASCAL Visual Object Classes Challenge Workshop, ECCV 2012, Florence, Italy, October 12. `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/workshop/history_analysis.pdf`.

Yudkowsky, Eliezer. 2013. *Intelligence Explosion Microeconomics.* Technical Report, 2013–1. Machine Intelligence Research Institute, Berkeley, CA. `http://intelligence.org/files/IEM.pdf`.