LEARNING FROM THE PROS

large-scale analysis of world-class solves reconstructions

FIRST EPISODE - CFOP

March 2021

REVISION V1

THE PEOPLE BEHIND THIS PROJECT

The following analysis relies on the concerted effort of a number of people

- **Stuart Clark**: the reconstruction god, single-handedly reconstructed more than a thousand of the solves that comprise the source data for this analysis; It is hard to render justice to the amount of effort (and speed) that went into reconstructing the solves whose features are synthesised here. Unable to stop there, Stuart was instrumental as a sparring partner in the analysis phase of this project

- **Gil Zussman**: the creator of speedcubedb.com, which among its many features collects and present all the reconstructions; Besides creating many of the tools that make reconstructions possible today, Gil's contribution in providing both data and insights was an essential part of this work

- **Basilio Noris:** obsessed with data visualisation, he plunged into the raw data and made this analysis and document, extracting what could be of interest and could provide new learnings and spent way too much time making colourful charts

A tribute to the original Recon God

While the vagaries of life sometimes force people to focus on new things, legacies remain. Brest not only reconstructed more than 2000 solves on his own, he trained and made the current generation of reconstructors what it is

A special thanks to all the solvers who have contributed their solves, sometimes having to suffer through our pleas for ao50s and ao100s, days or weeks on end. Even if all your contributions have not borne fruit yet, they are at the heart of what it has been possible to do here. And a final thanks to the **Reconstruction Friends** discord, which – besides fostering a culture of exchange and sharing – reunites most of the efforts of reconstruction that has allowed this analysis to exist. And a final thanks to Feliks, Phillip and Ben for reviewing this in its final phase of preparation.

Cover Image: I haven't been able to find the original photographer to give proper credits, if you know where this comes from you're welcome to tell me or put me in contact with the photographer!

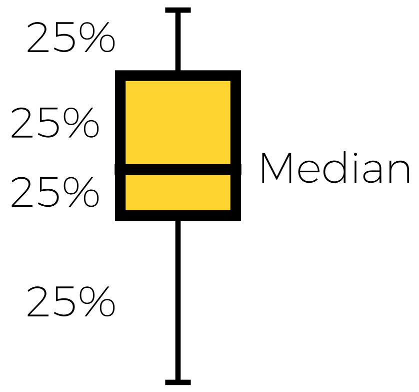
HOW THIS DOCUMENT IS STRUCTURED

THE ANALYSIS IN ITS DIFFERENT PARTS

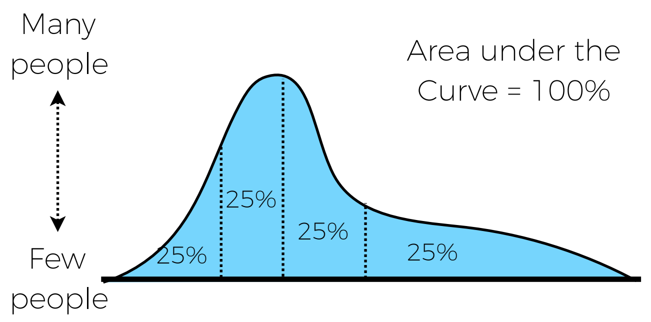
- **The Dataset:** A quick overview of the data and its features (and flaws)
- **Solve-level analysis**: what can we learn from solves of the fastest solvers? What elements are common to all people, which seem peculiar to some but not all?
- **CFOP: All we can learn about cross:** are there common elements to efficient crosses (e.g. 2-3-4gen)? What about rotations? Do x- and xxcrosses come with a certain frequency, and are they really worth it?
- **CFOP: First two layers:** is there a core of "frequent pairs" that get selected early in the solve (1st/2nd slots)? What are the preferred inserts, and do they change significantly across solvers? Rotations vs fancy executions, is there a clear consensus?
- **CFOP: Last layer:** what can we learn from last layer execution? Are zbll algs worth the recognition slowdown? How often are skips happening? How much of that is due to influencing vs chance?
- **Conclusions and moving forward:** Many things remain to be done, least of which is tackling the other methods (Roux, I'm looking at you!)

A VERY SHORT CRASH COURSE ON DATA VISUALISATION

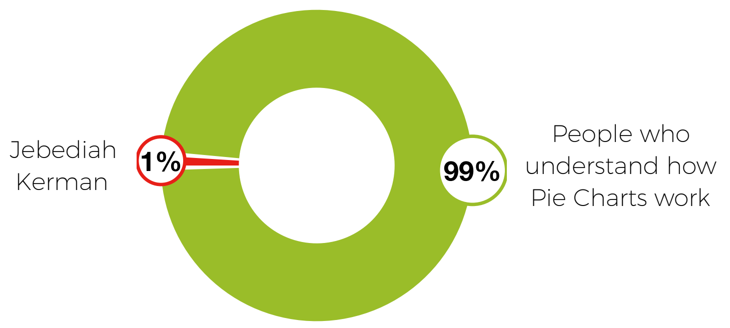
Box Plots



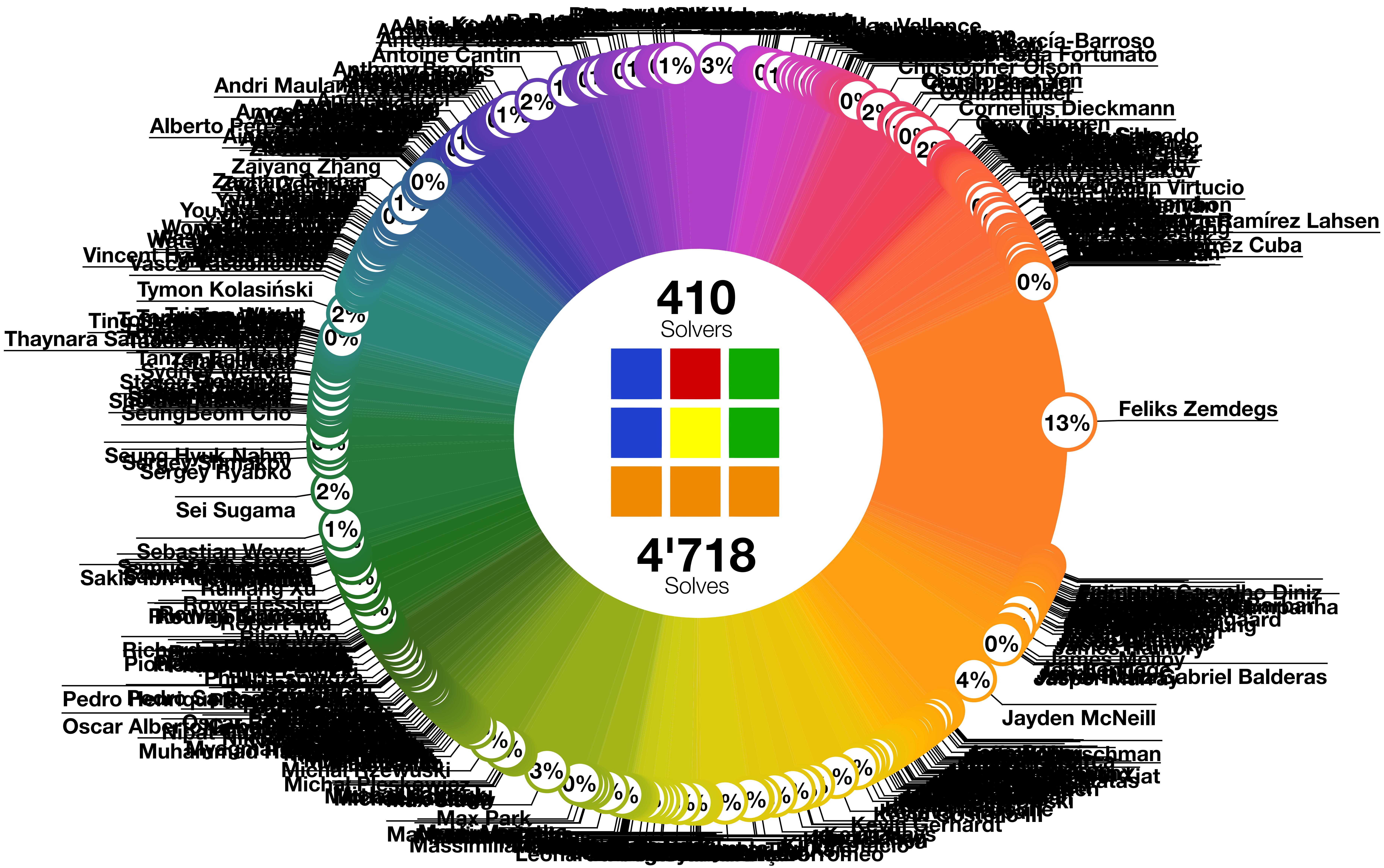
Density Estimator



Pie Charts



A BIRD'S EYE VIEW ON THE DATA AT THE TIME OF WRITING



THE DATASET IN NUMBERS

4'718 solves

410 solvers

5 solves each (median)

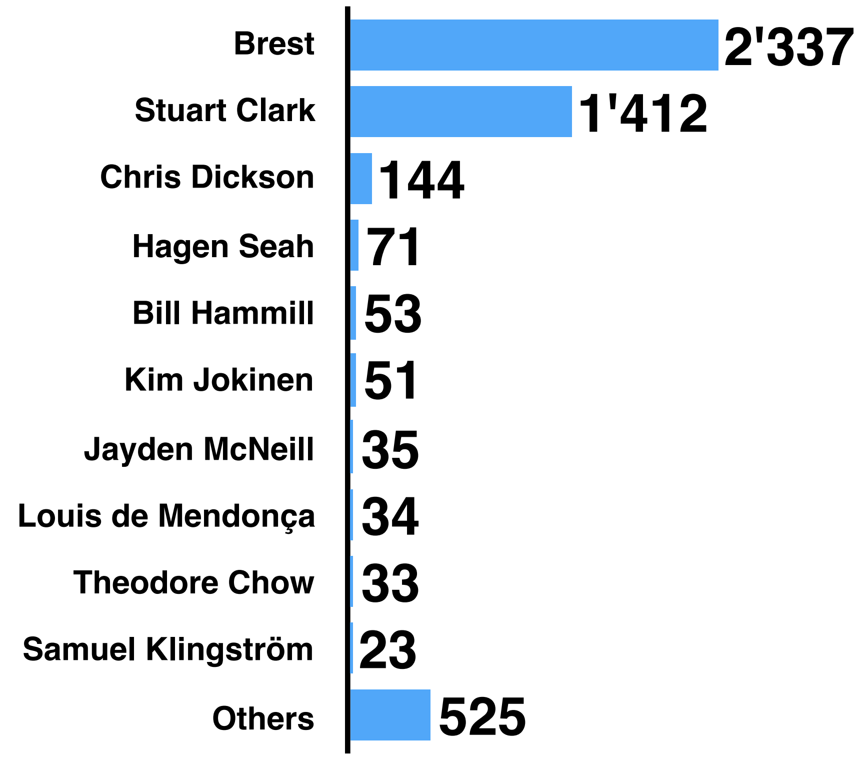
12h of solve time

8.16 median solve

252'916 moves

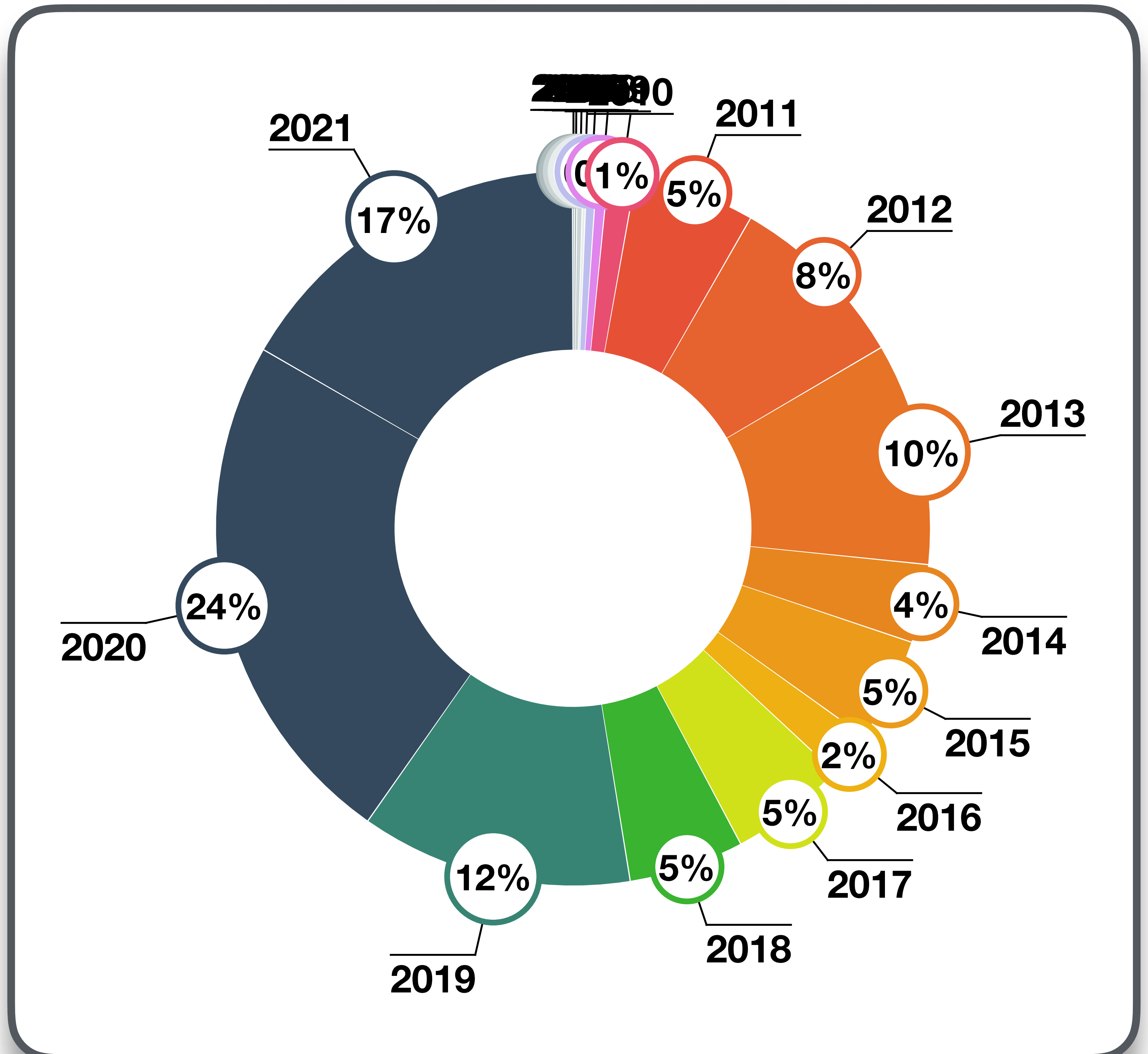
1 website

150 reconstructors



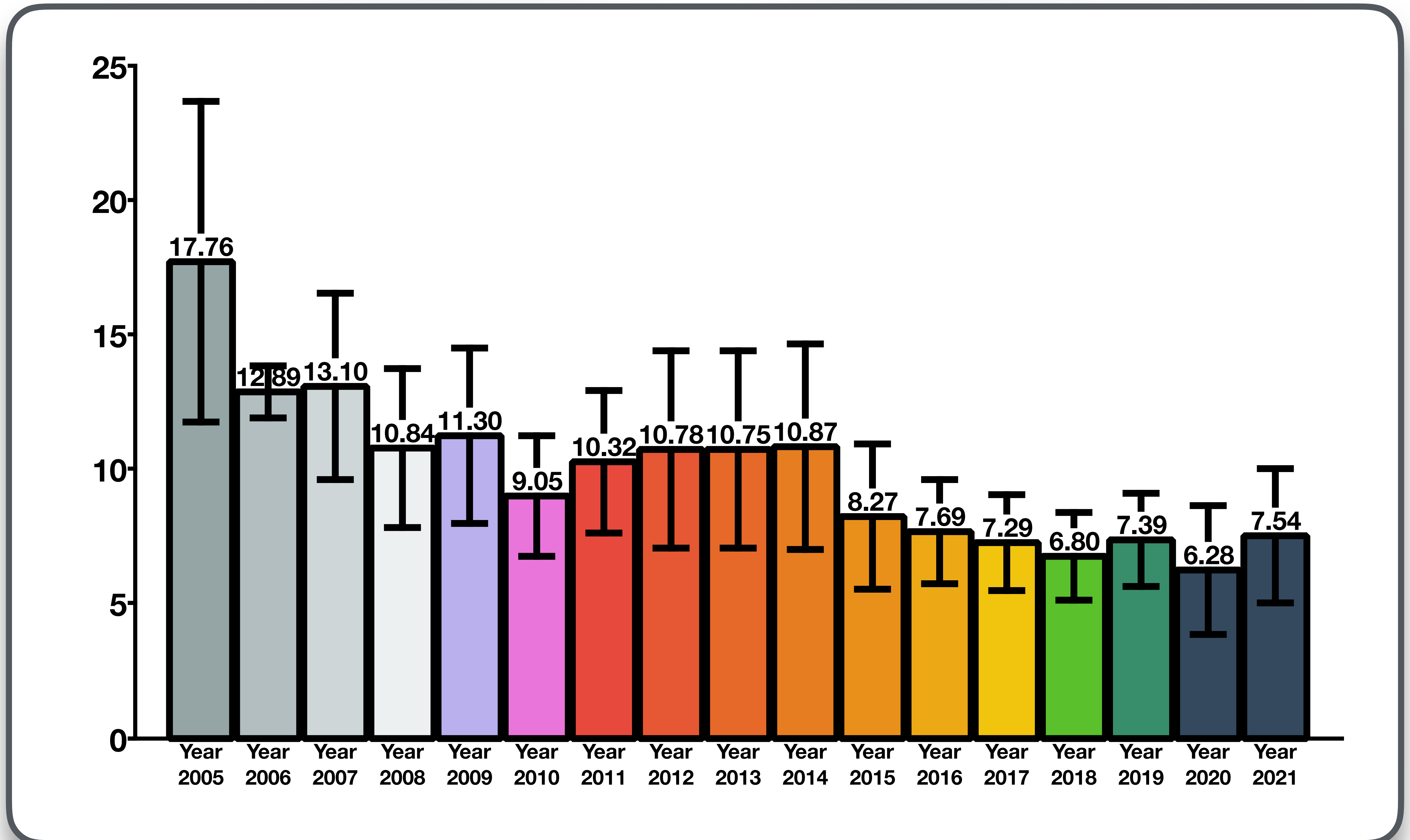
SOLVES SPAN A LONG PERIOD OF TIME (WITH A BOOST IN THE LAST 3 YEARS). AS WE ALREADY KNOW, TIMES HAVE SHRUNK CONSIDERABLY OVERALL AS HARDWARE AND "SOFTWARE" HAVE GOTTEN BETTER

DISTRIBUTION OF SOLVE YEARS



Note: Excludes solves for which a date is not available (~41%)

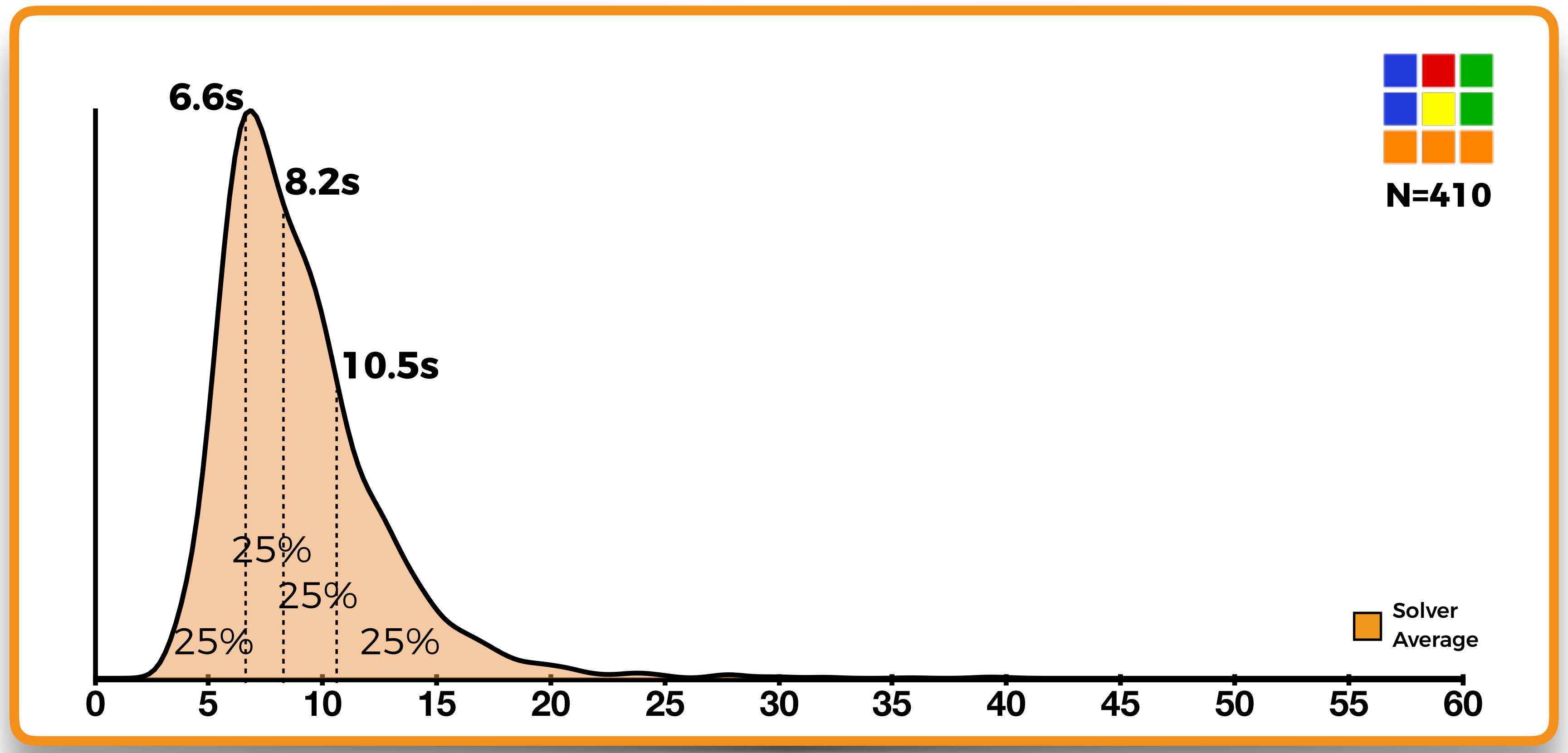
SOLVE TIME BY YEAR



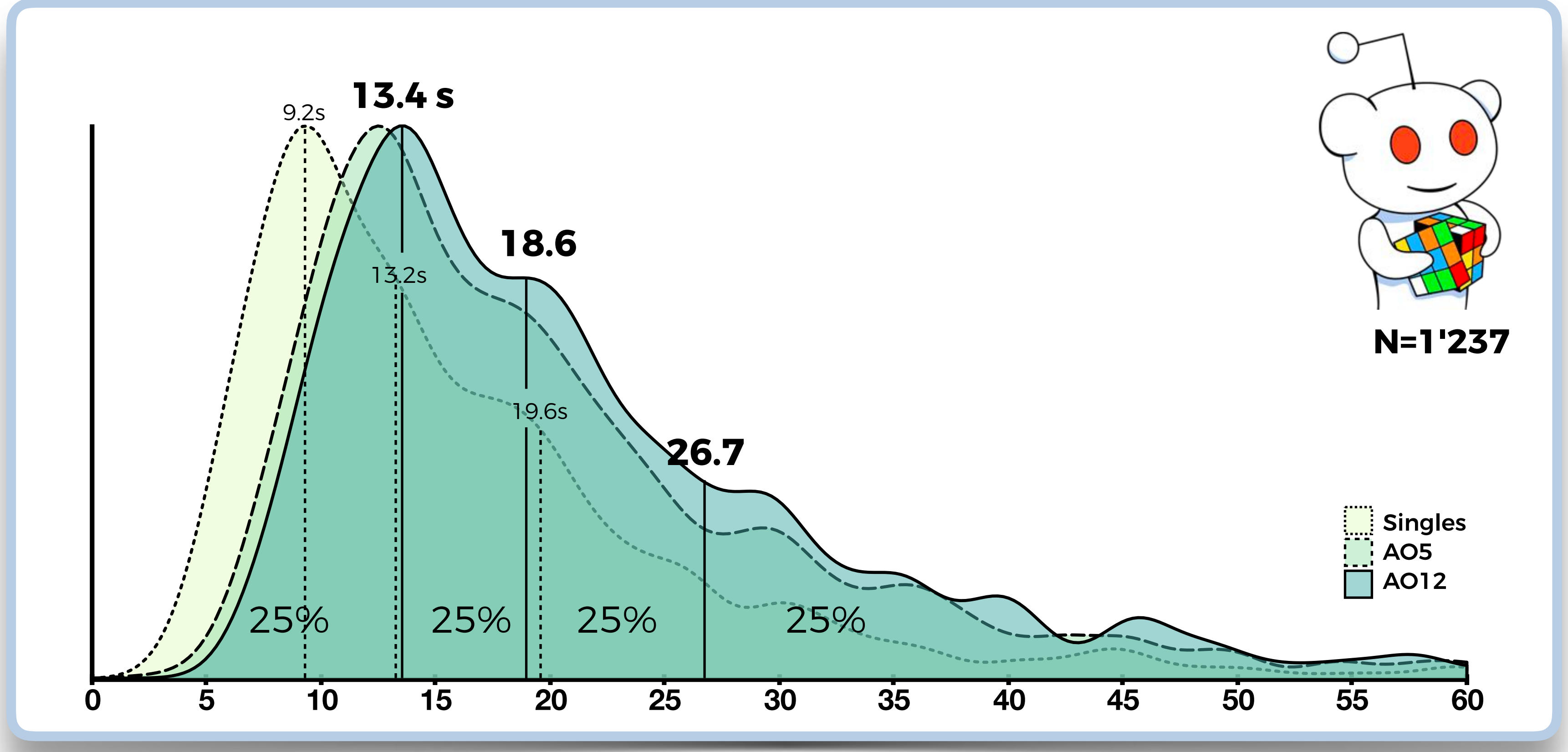
BIASES IN THE DATA, AND IN THE ANALYSIS

- **Solves:** partly by design a majority of the reconstructions here are very good (and maybe very lucky) solves. This means we are not always encapsulating what would happen with "great cubers, nasty scrambles", and whether specific strategies might work better than others on these.
- **Speedcuber-level analysis:** we don't have the same amount of data from all speedcubers, for some we have 50 solves, for others 5, for a couple we have hundreds. This means that we sometimes only have a selection of the very best solves, rather than an overall understanding of the habits and solving particularities of the speedcubers themselves.

3X3 AVERAGES FOR SPEEDCUBEDB SOLVERS

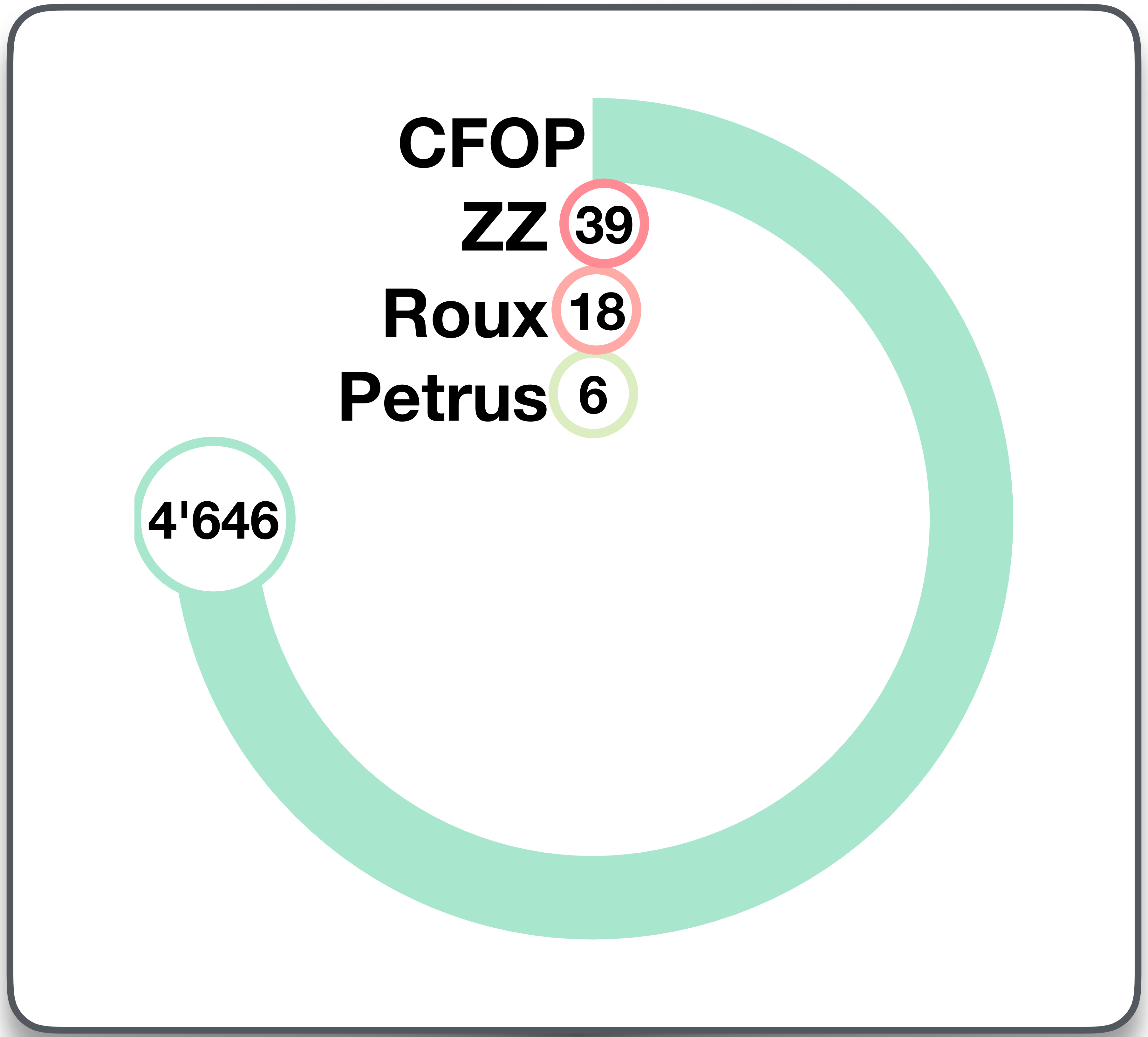


3X3 SOLVE TIMES FOR /R/CUBERS SOLVERS (2020)



A BIG BIAS IN THE AVAILABILITY OF DATA WILL FOCUS OUR ANALYSIS ON CFOP

DISTRIBUTION OF SOLVES BY METHOD



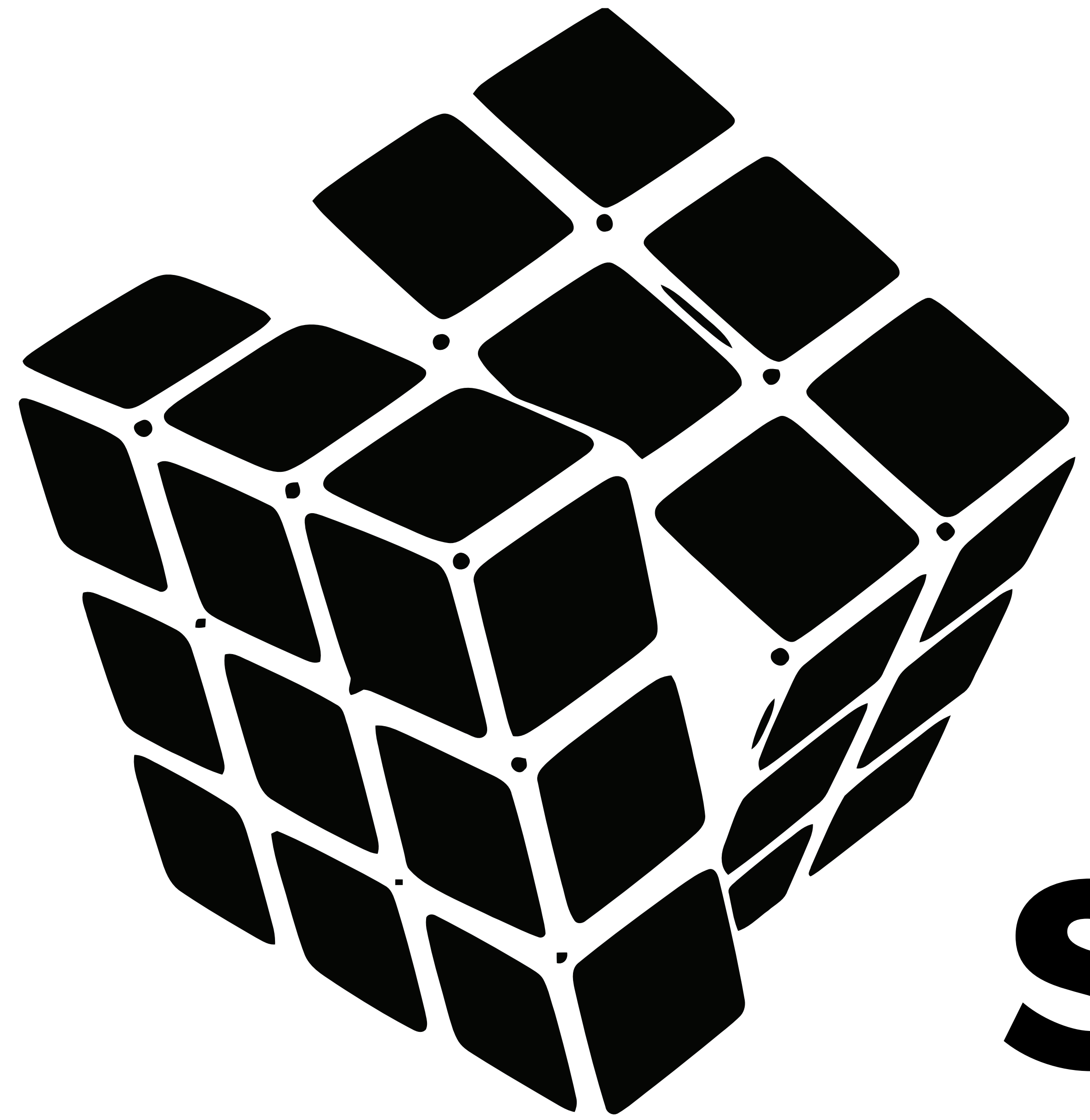
- **The eternal battle of the big 4, or big 2, or big whatever:** When the database started, a focus was understandably put on the prevalent method, and on the fastest solves, which happened to coincide in the CFOP / Fridrich method. This is not to say that the other methods do not provide plenty of material for insightful understanding of what makes solving the cube possible, but we simply don't have enough data on those (yet) to obtain reliable results

This is why most of this analysis (for now) will revolve around the CFOP steps

AND FINALLY A WORD OF CAUTION

"I do think world class F2L (and now even LL) is half art, half science though, and fingertricks/regrips are such a key element."
– Feliks Zemdegs

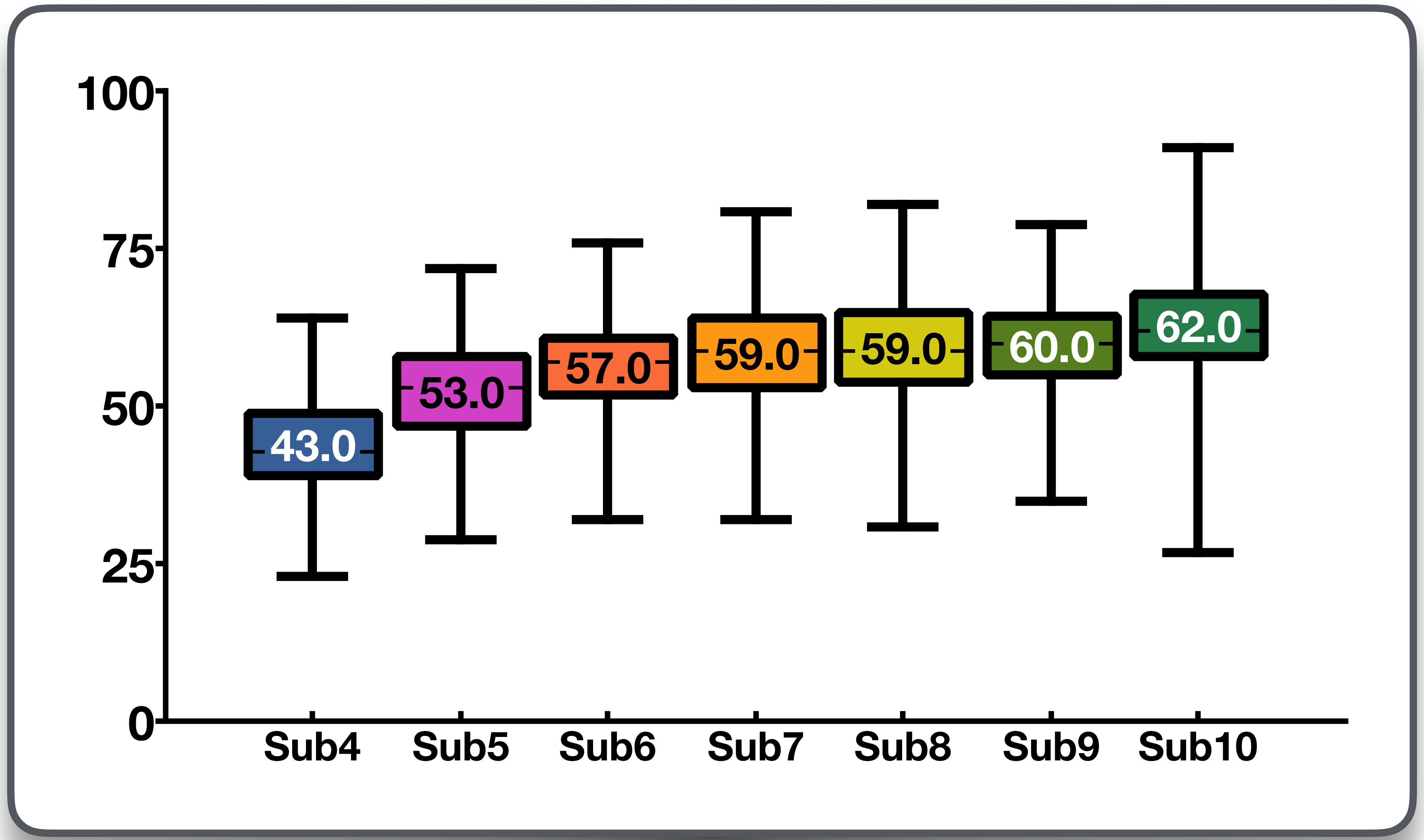
So let's not take all of this too seriously!



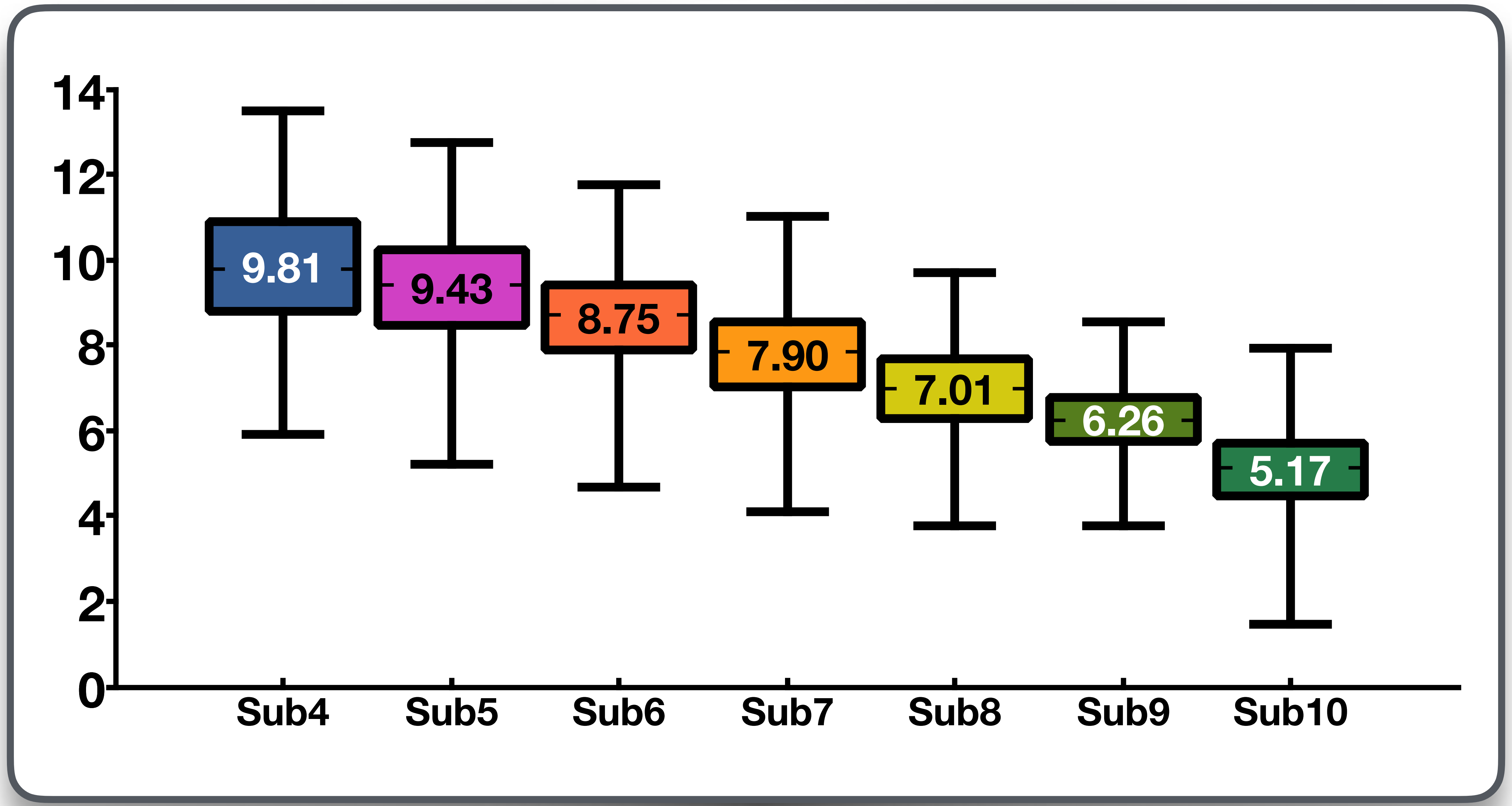
CHAPTER 1 : THE SOLVES AND SOLVERS, OVERALL

UNSURPRISINGLY, THE FASTER THE SOLVE, THE LOWER THE MOVE COUNT AND THE FASTER THE TPS, BUT THE TWO DO NOT HAVE TO GO IN LOCKSTEP

GENERAL MOVES BY SOLVE TIME



GENERAL TPS BY SOLVE TIME

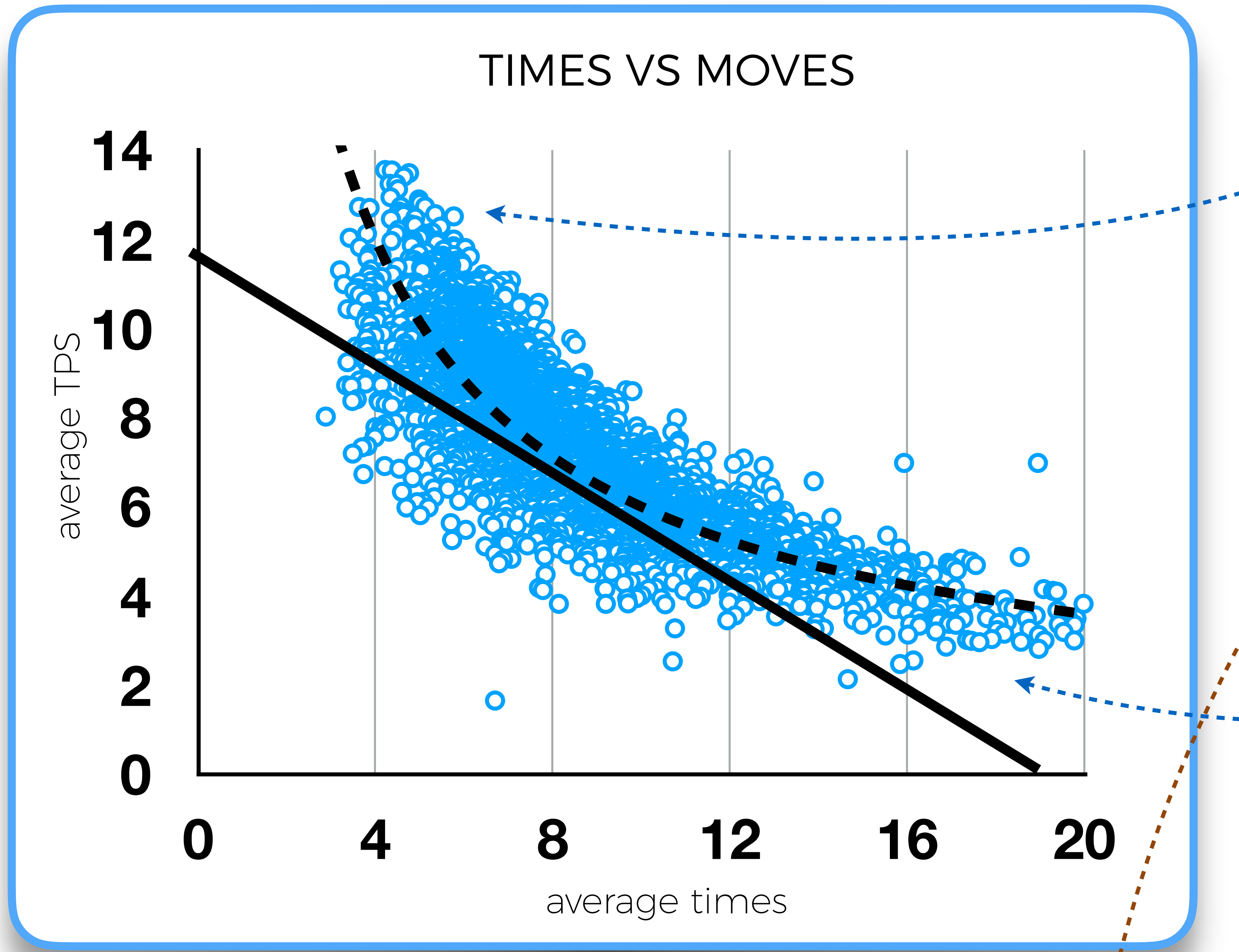


The chaos at the edges

While on average faster solves have fewer moves, there are exceptions, with 50-60+ move solves managing to be Sub4, conversely, an insane TPS does not always means the faster time

N=4000+

TPS VS EFFICIENCY : THERE SEEMS TO BE A TRADE-OFF AFTER SOME POINT



Diminishing returns

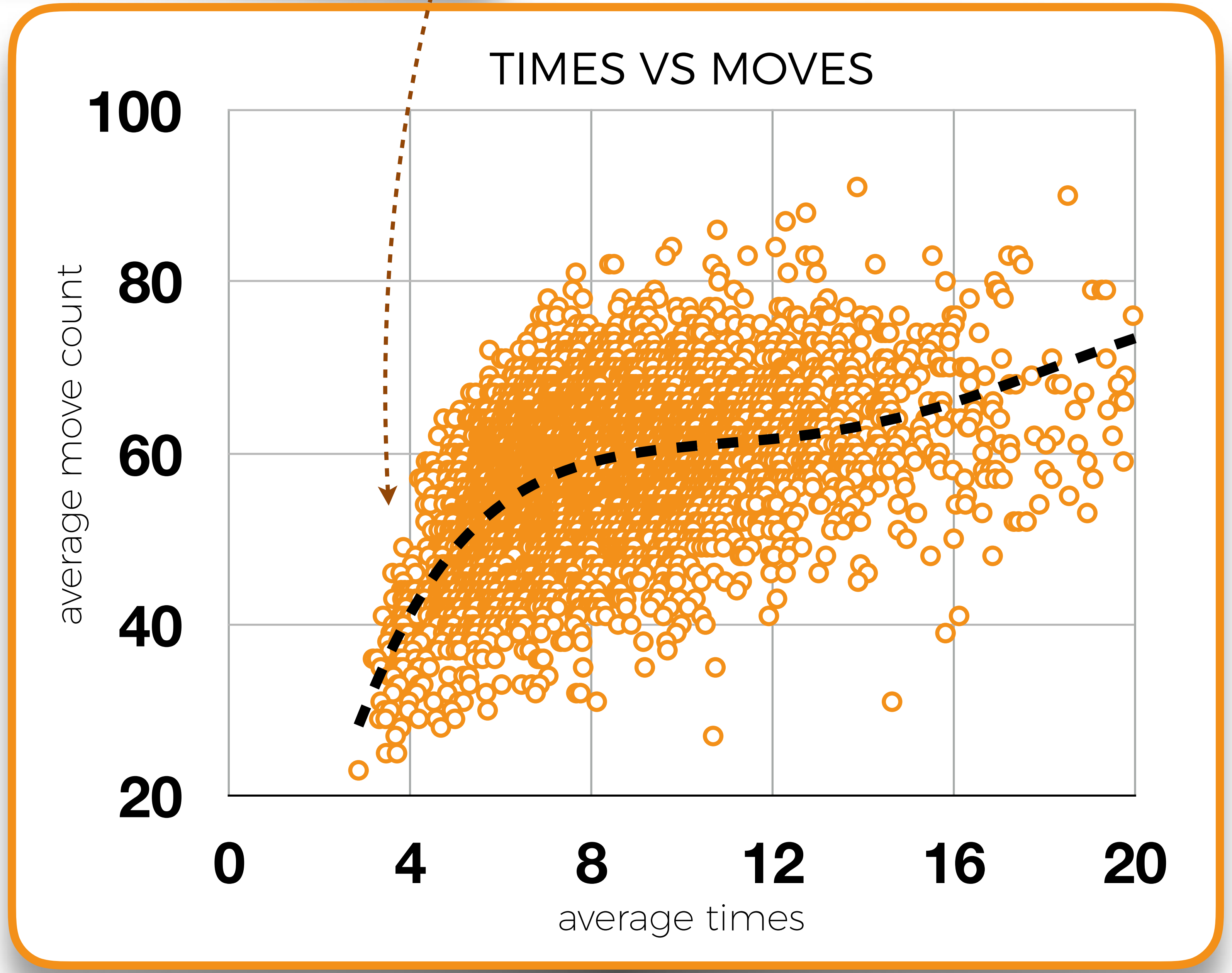
Above a certain TPS, solvers are unable to ensure the same degree of efficiency to proportionally reduce their times

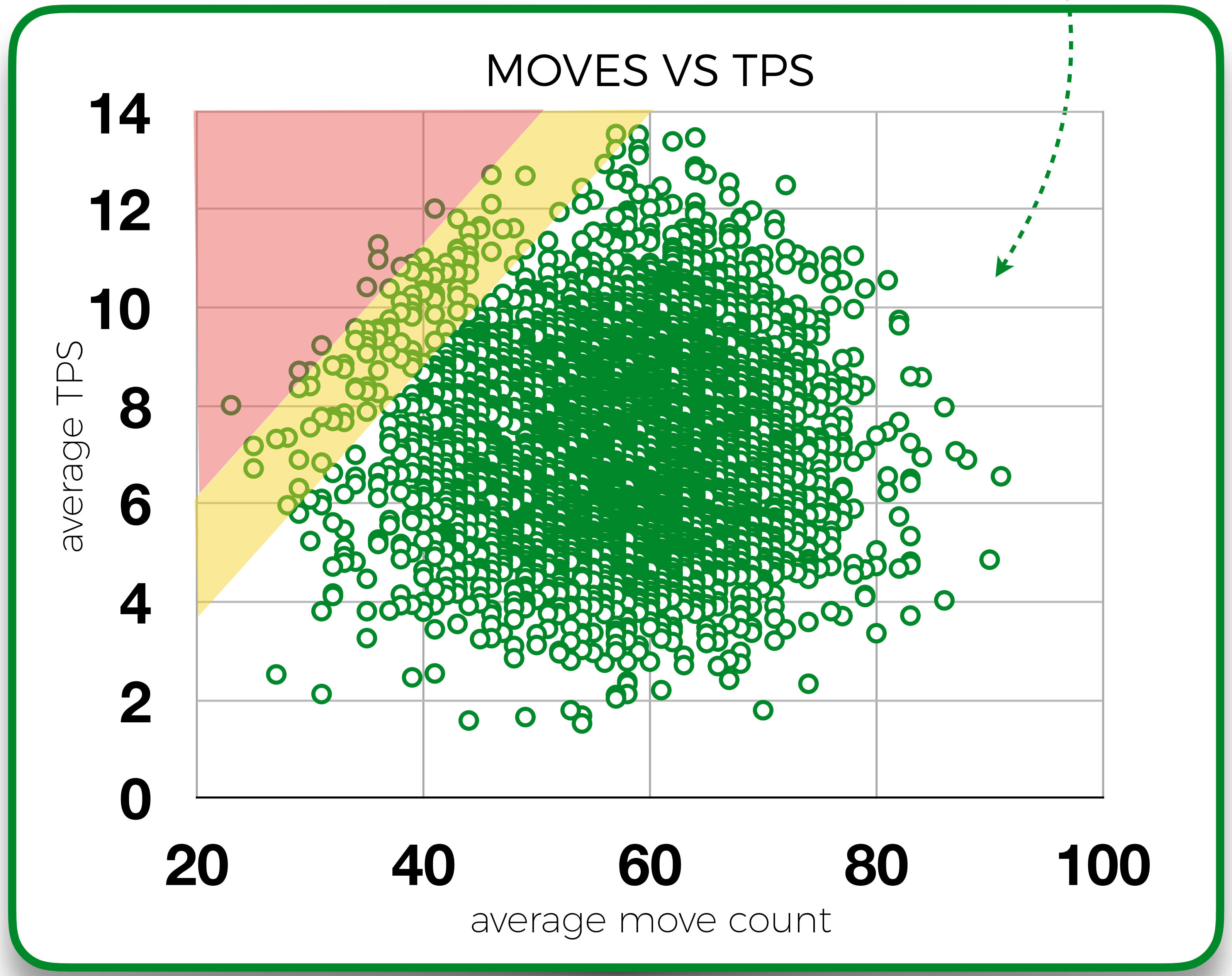
Efficiency Gap

The performance of slower speedcubers is less a function of TPS, and more a lack of move efficiency

Not much of a link

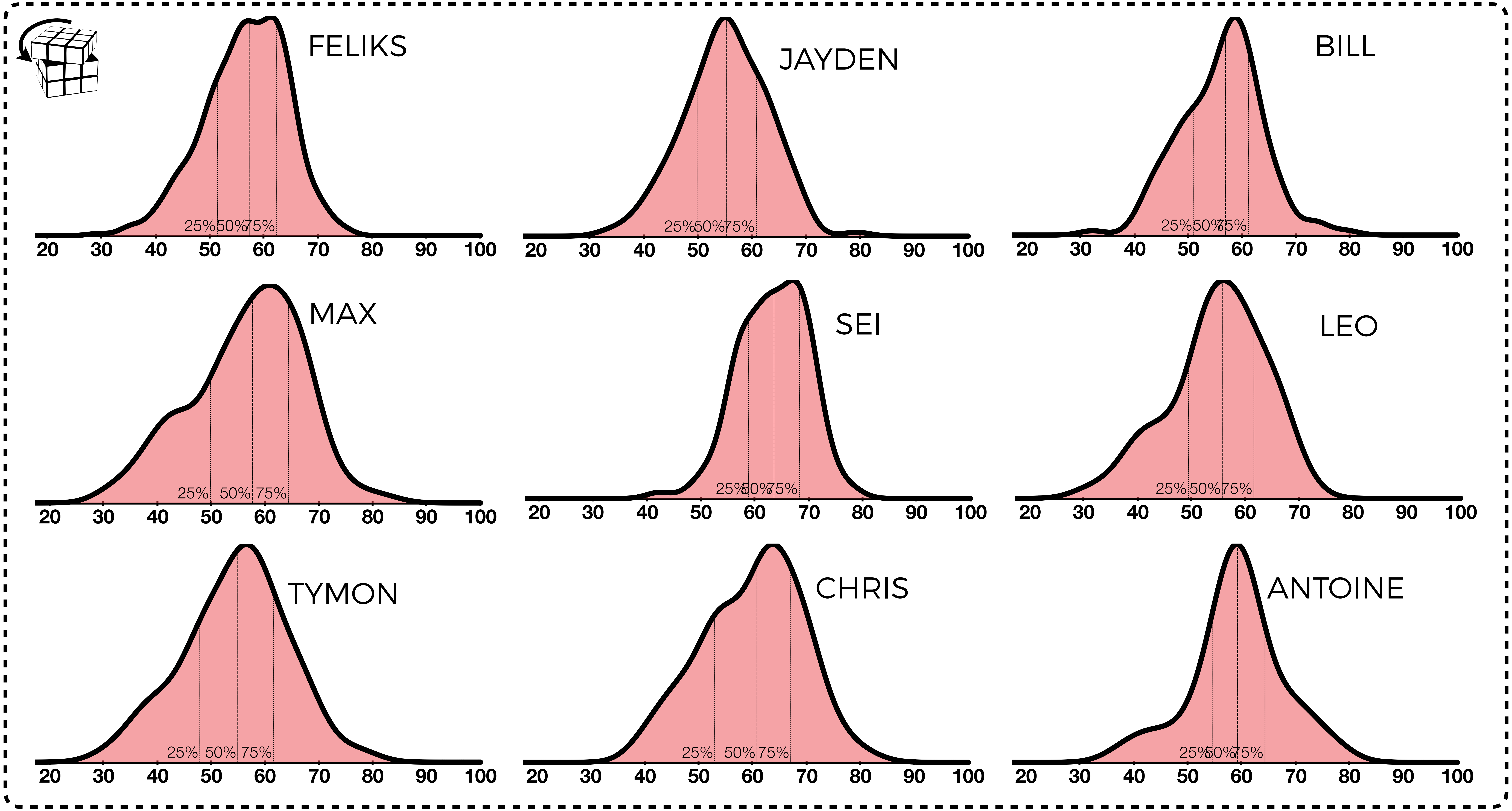
"Everyone can be efficient": There is little correlation between average TPS and move count





N=4000+

VERY DIFFERENT SPREADS IN THE NUMBER OF MOVES FOR DIFFERENT SOLVERS



N=1541

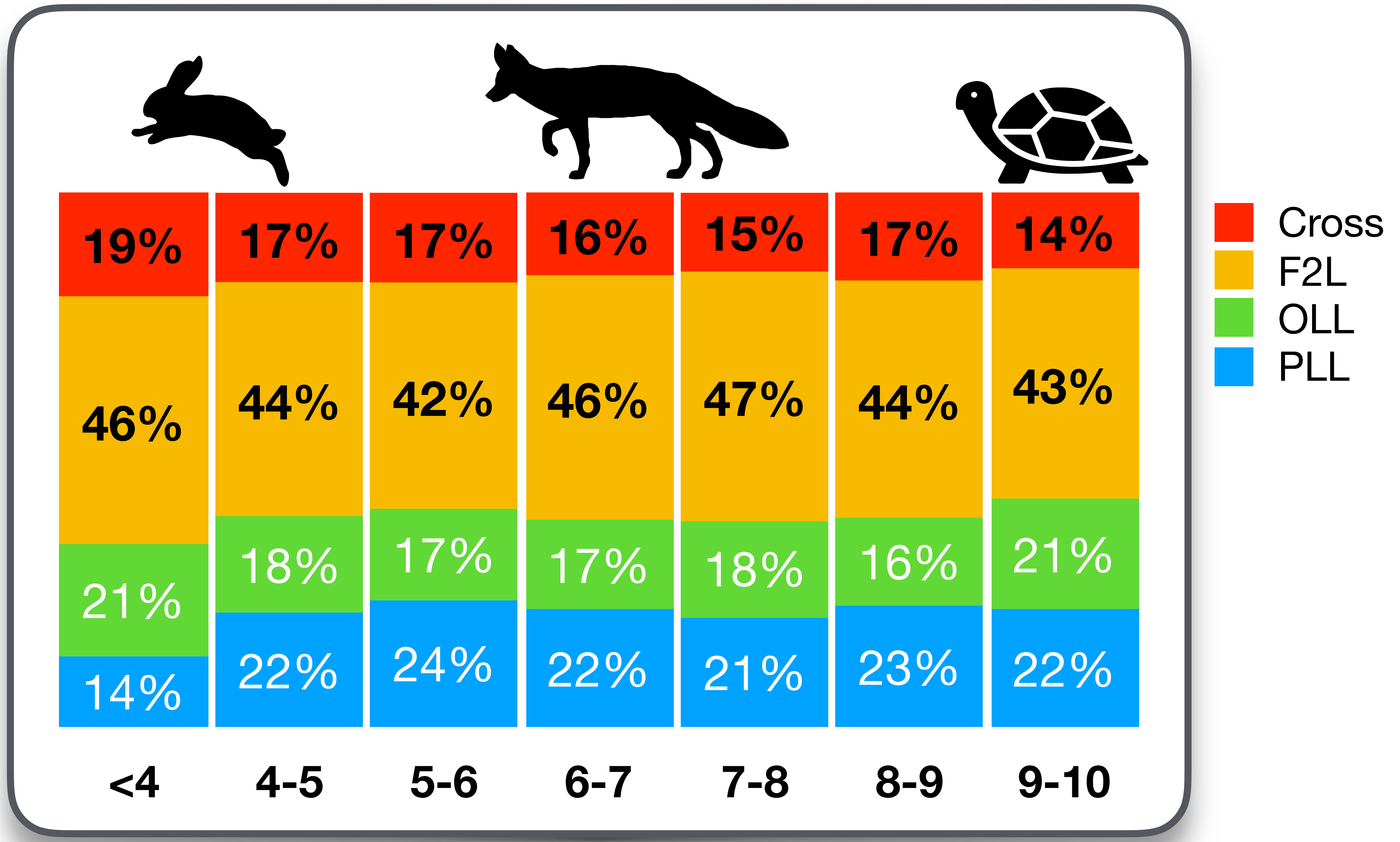
9 Solvers for whom the most solves are available

AT THE FASTEST LEVEL OF SOLVES, LAST LAYER SHRINKS (THANKS TO SKIPS), AND CROSS TAKES UP A BIT MORE OF THE SOLVE TIME (DUE TO XCROSSES)

TIME SPLITS SUB4 VS SUB10

	SUB4	SUB10
CROSS	**19%** 0.87s	**14%** 1.50s
F2L	**46%** 1.97s	**43%** 4.40s
LL	**35%** 1.86s	**43%** 3.30s

PROGRESSION OF TIME SPLITS



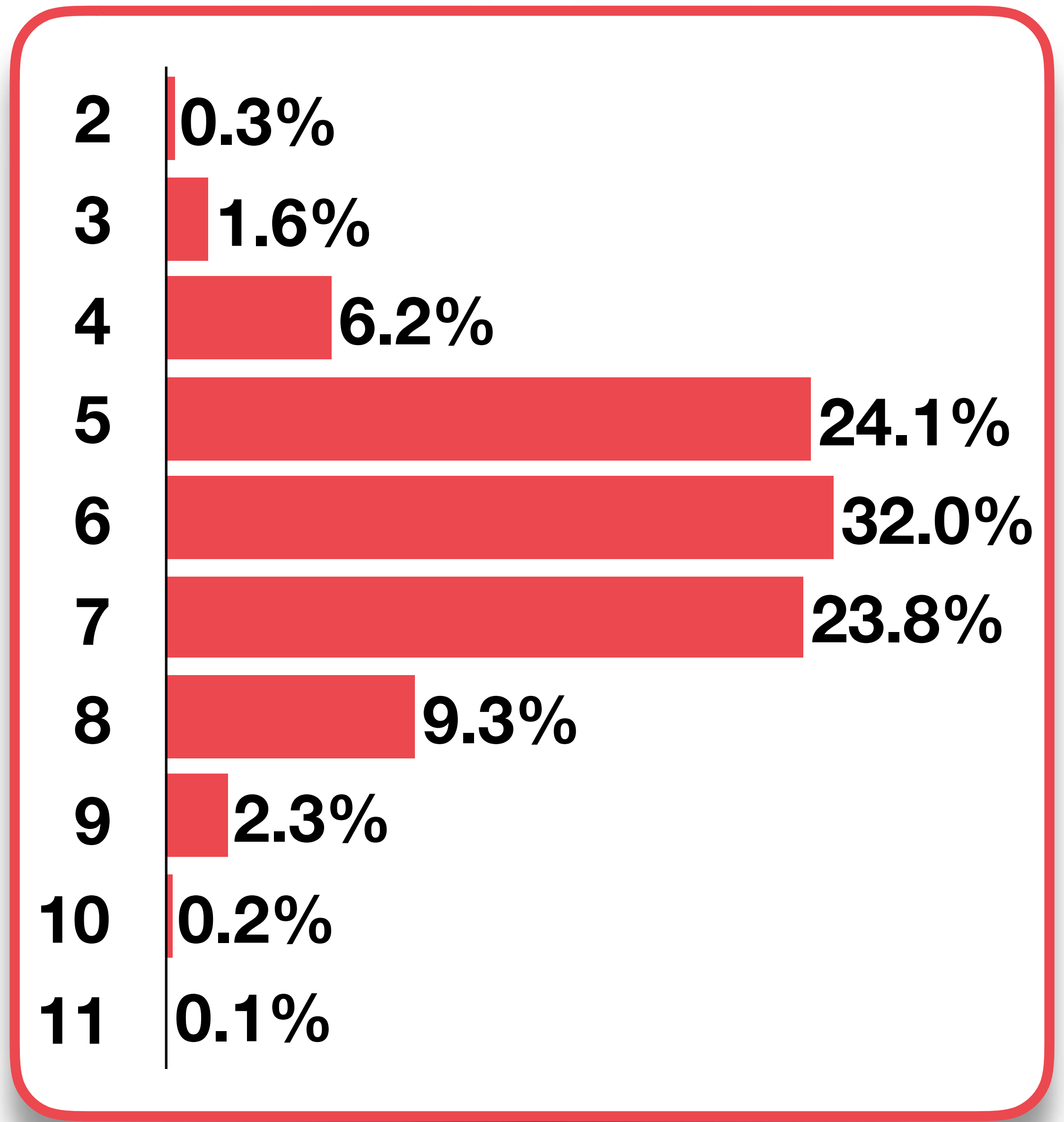
Cross is NOT getting longer

While the share of total solve time for cross goes up, the absolute time of Cross and f2l drop as well, they simply drop less than last layer

N=830+

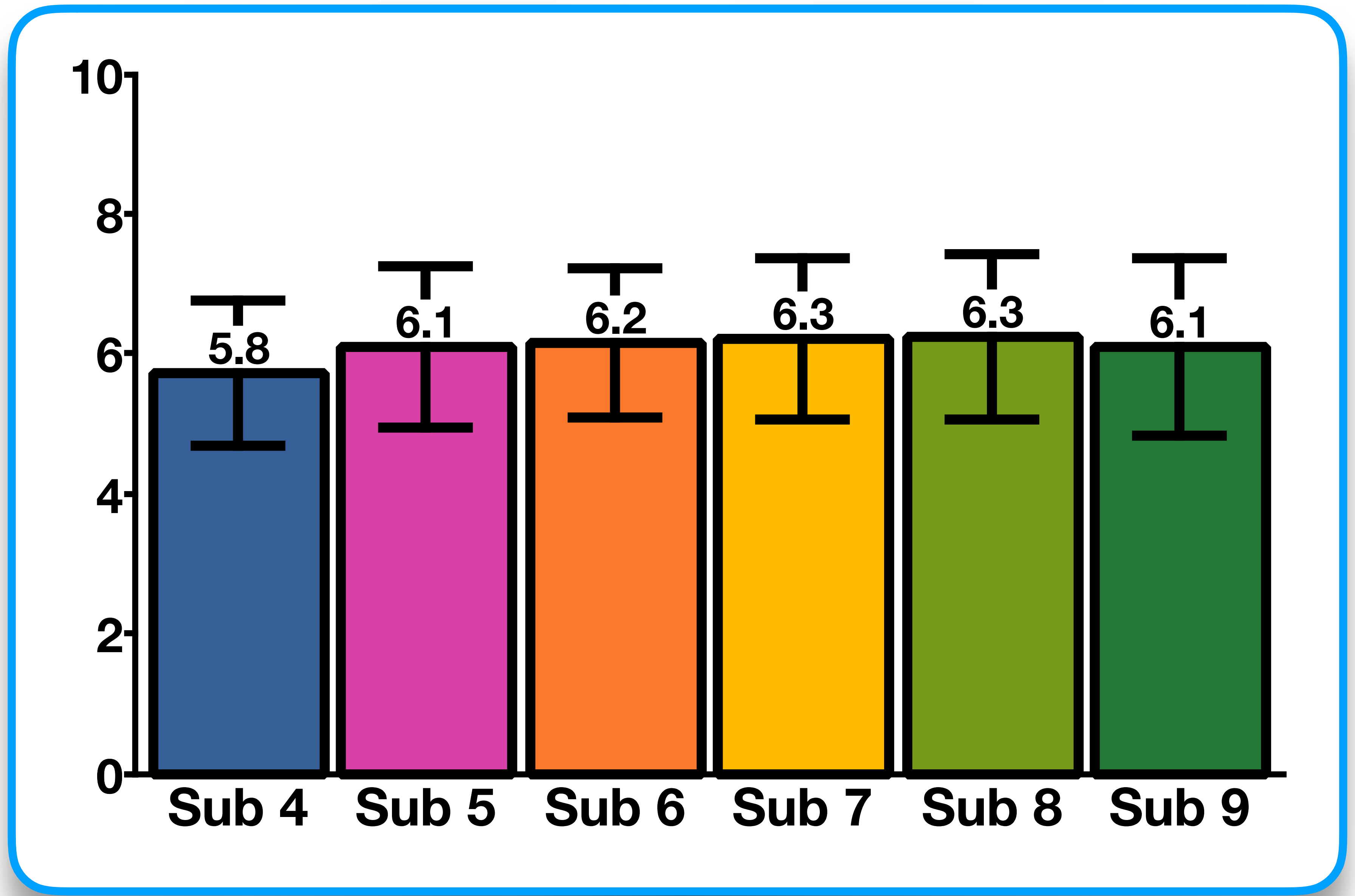
MOST SOLVES ARE BETWEEN 5 AND 7 GEN, WITH CROSS BEING THE MOST COMPLEX STEP

TOTAL SOLVE GEN



Average Gen:
6.1 gen

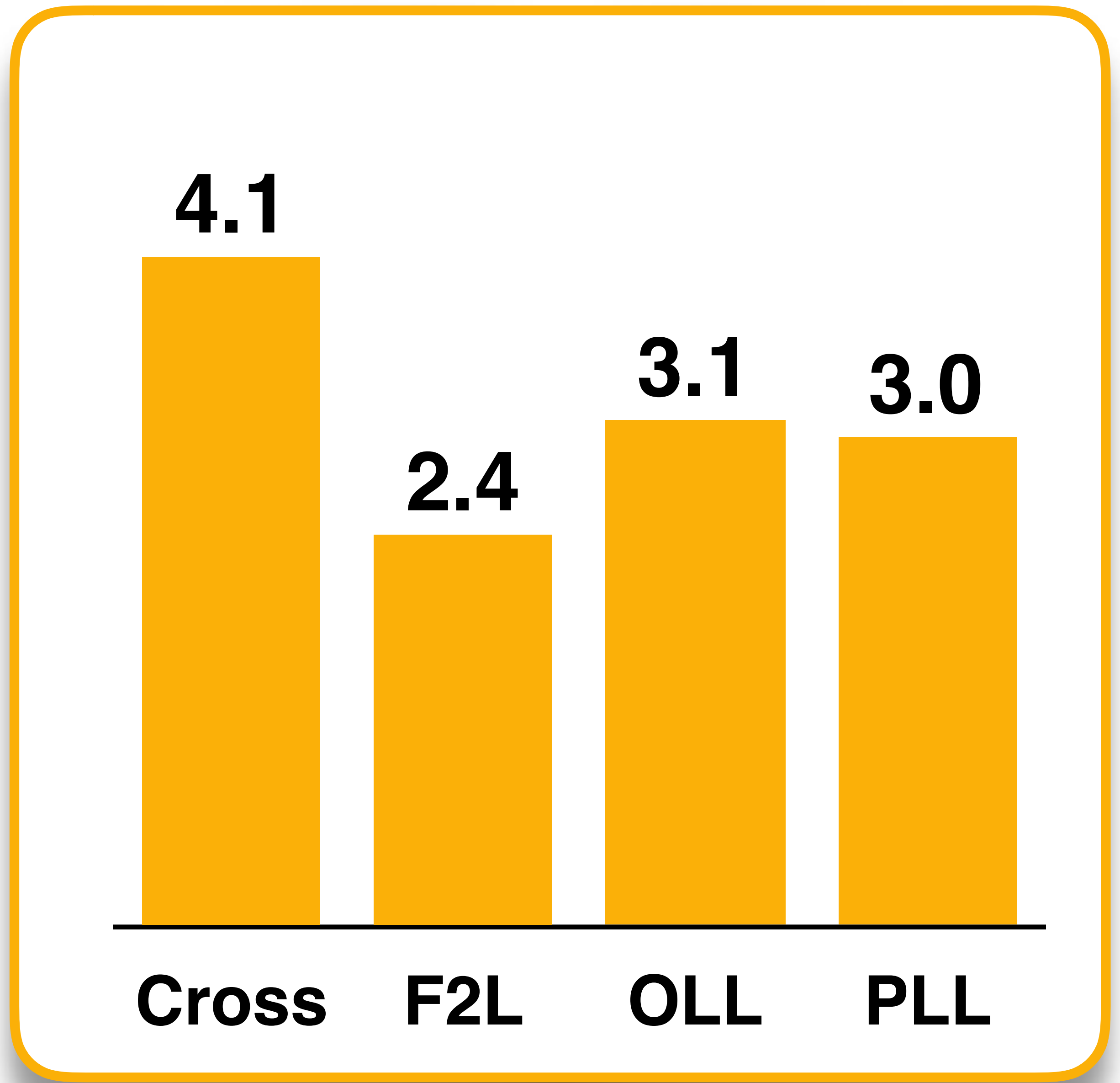
AVERAGE GEN BY SOLVE TIME



It's not really the gen

Gen choice does not seem to be affecting times, with no difference between high vs low times (slightly lower for sub4, but not significantly so)

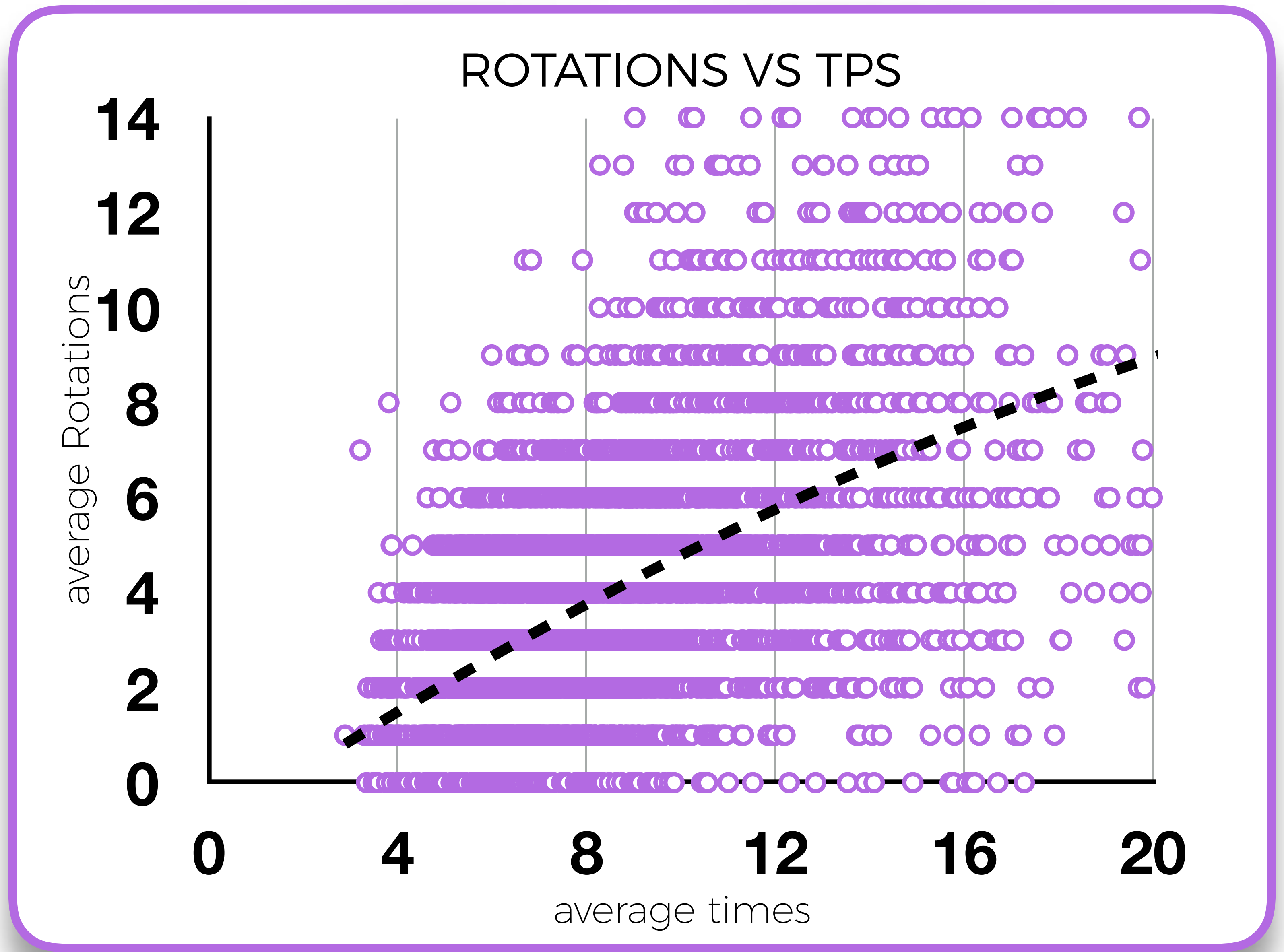
AVERAGE GEN PER STEP



Avg. Step Gen:
2.8 gen

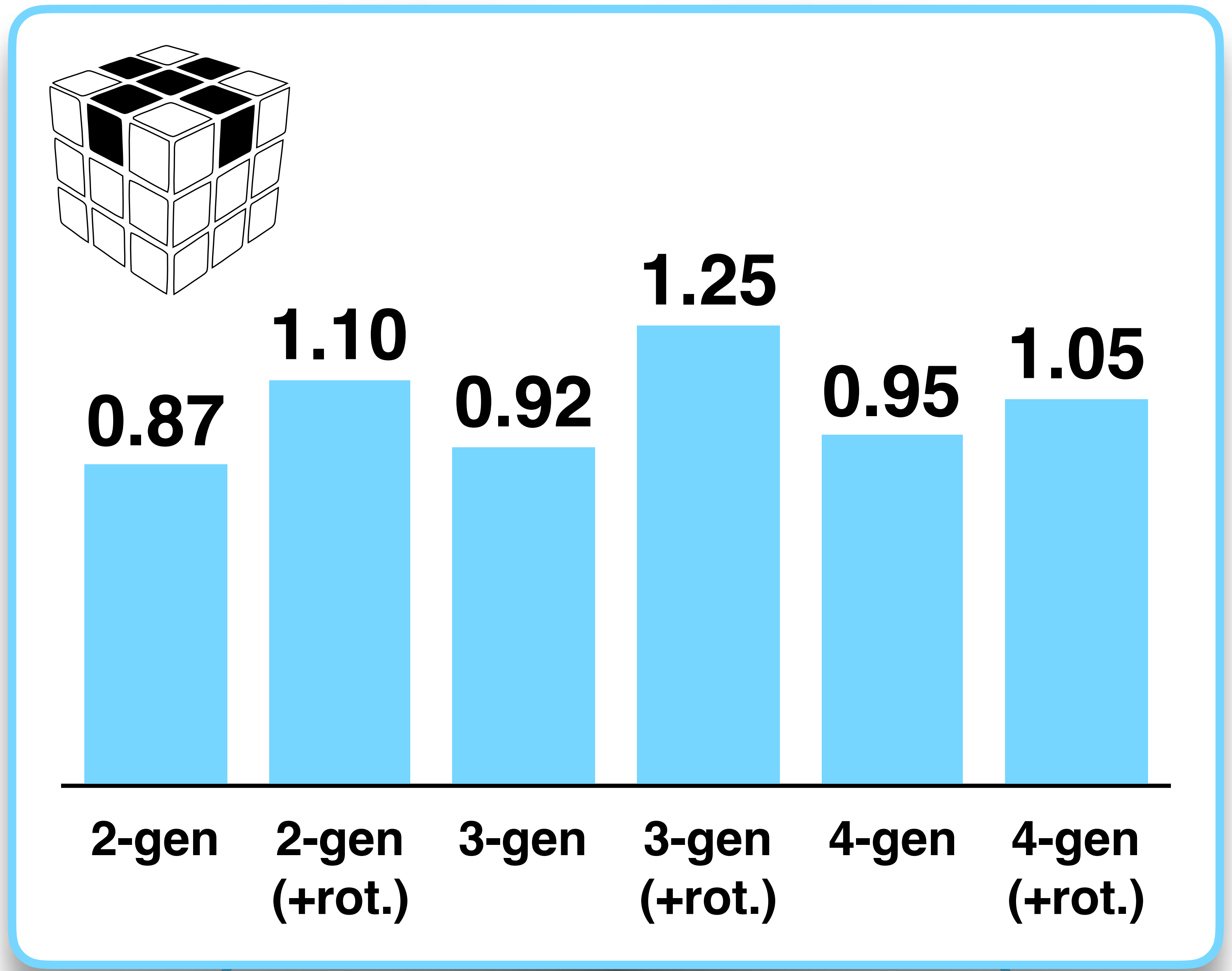
N=4000+

ROTATIONS VS GEN : NEVER ROTATE DURING CROSS, ALWAYS ROTATE FOR F2L!



CROSS EXECUTION TIME BY N-GEN AND ROTATIONS

For 5- to 7-move crosses

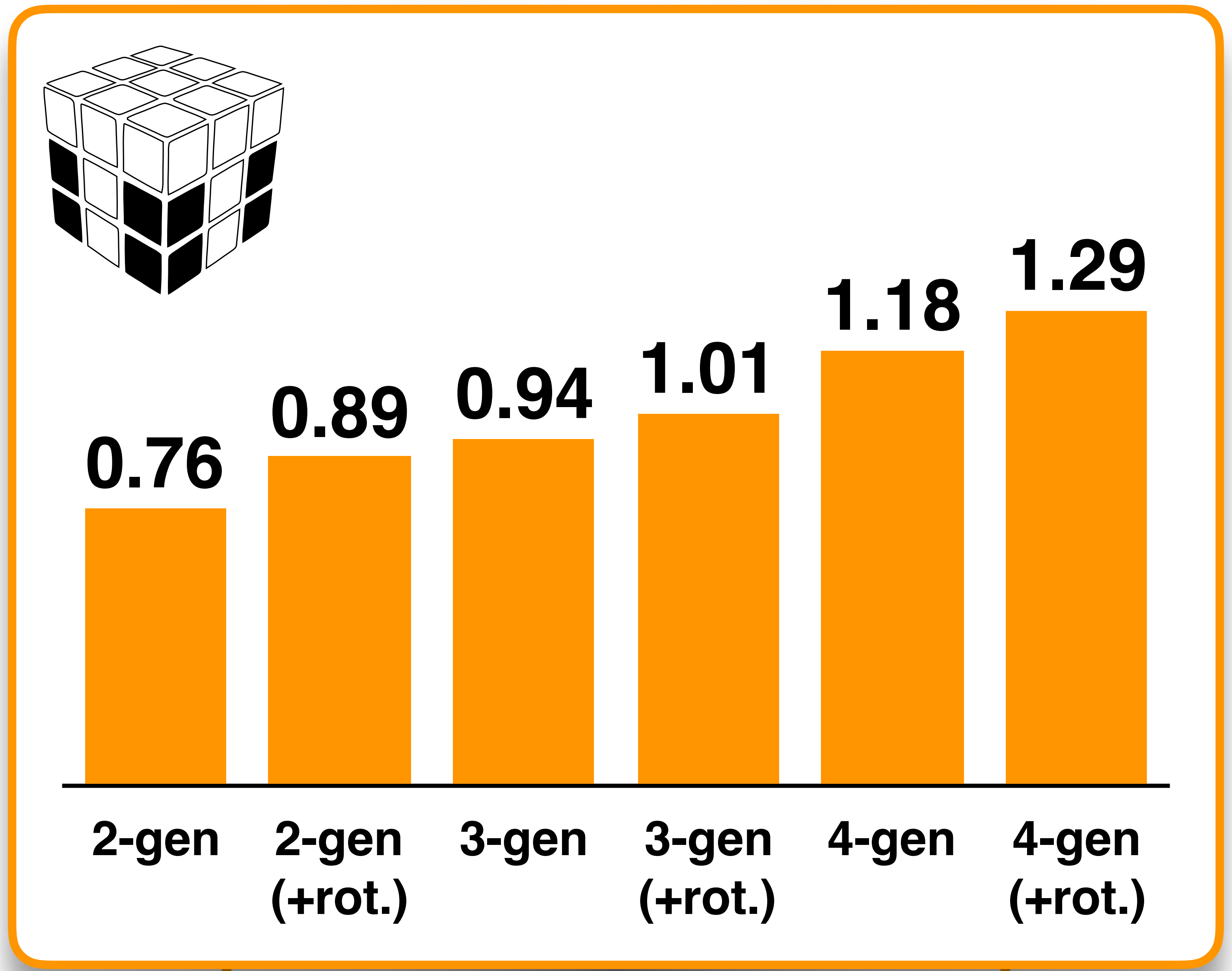


AVG TIME LOSS OF ADDING

1-Gen	Rotations
0.04 sec	**0.22** sec

F2L PAIR EXECUTION TIME BY N-GEN AND ROTATIONS

Average of all 4 F2l Pairs



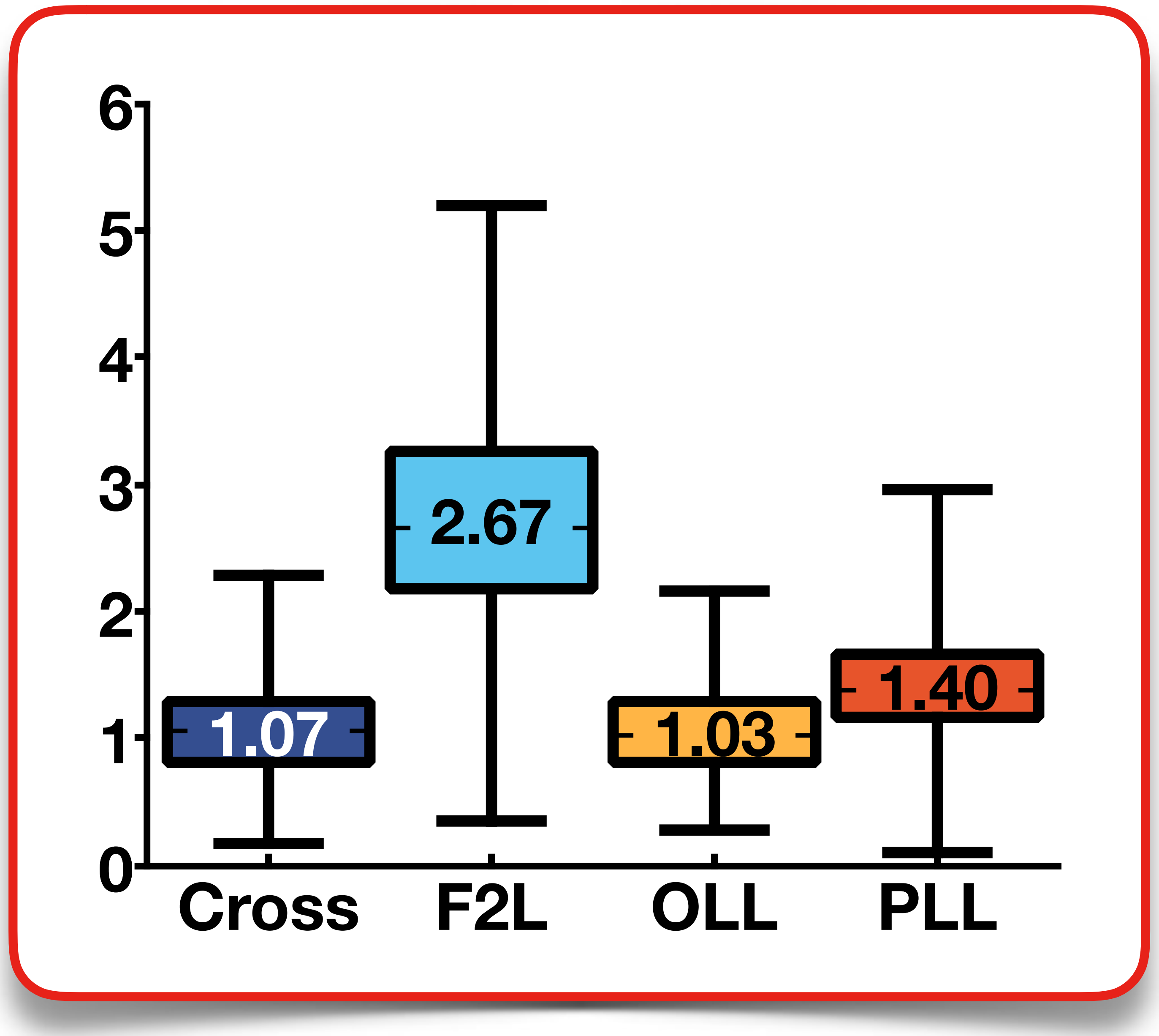
AVG TIME LOSS OF ADDING

1-Gen	Rotations
0.21 sec	**0.10** sec

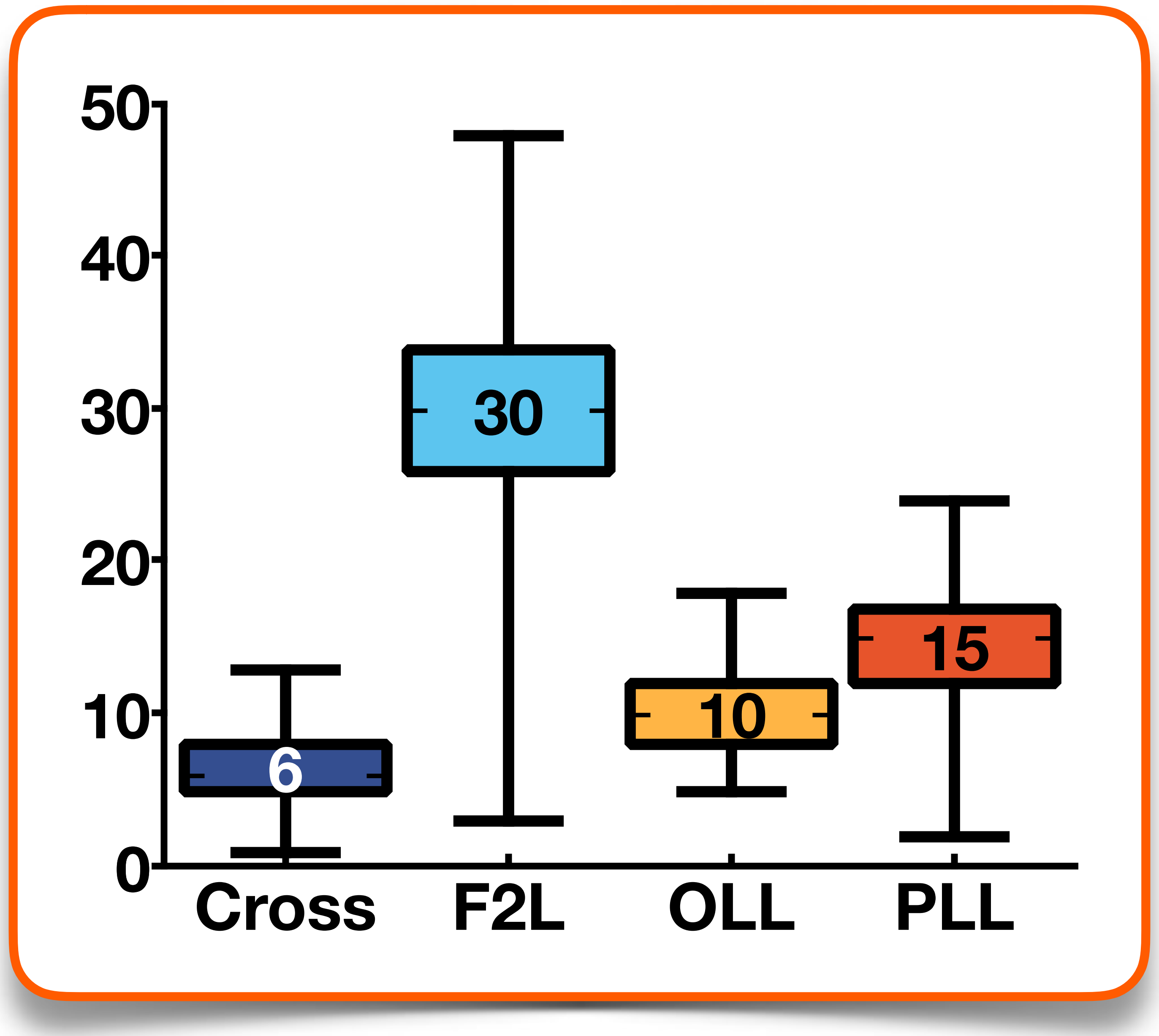
N=4000+

F2L TAKES UP THE LARGEST PART OF THE SOLVE TIME AND MOVE COUNT, BUT IS PERFORMED WITH PRETTY HIGH TPS, CROSS IS THE ONE THAT USES THE QUIRKIEST MOVES, AND IS PERFORMED AT A LOWER TPS

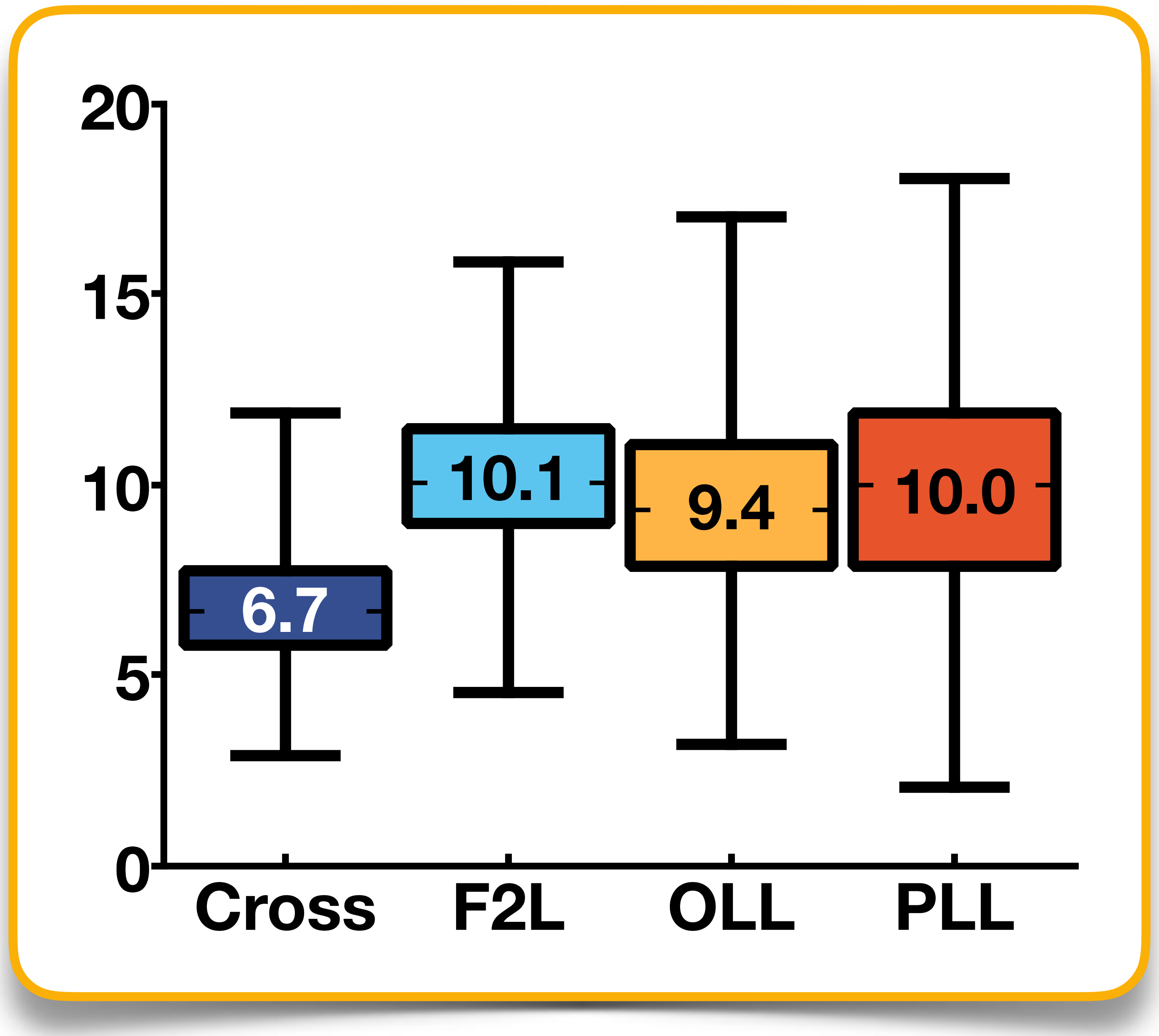
CFOP STEP EXEC. TIME



CFOP STEP MOVE COUNT



CFOP STEP TPS



When they say it's all about f2l

The largest variation in CFOP solves comes down to f2l: this is what makes or breaks a solve. However the other steps should not be discounted, as every bit helps (or hurts) the overall results

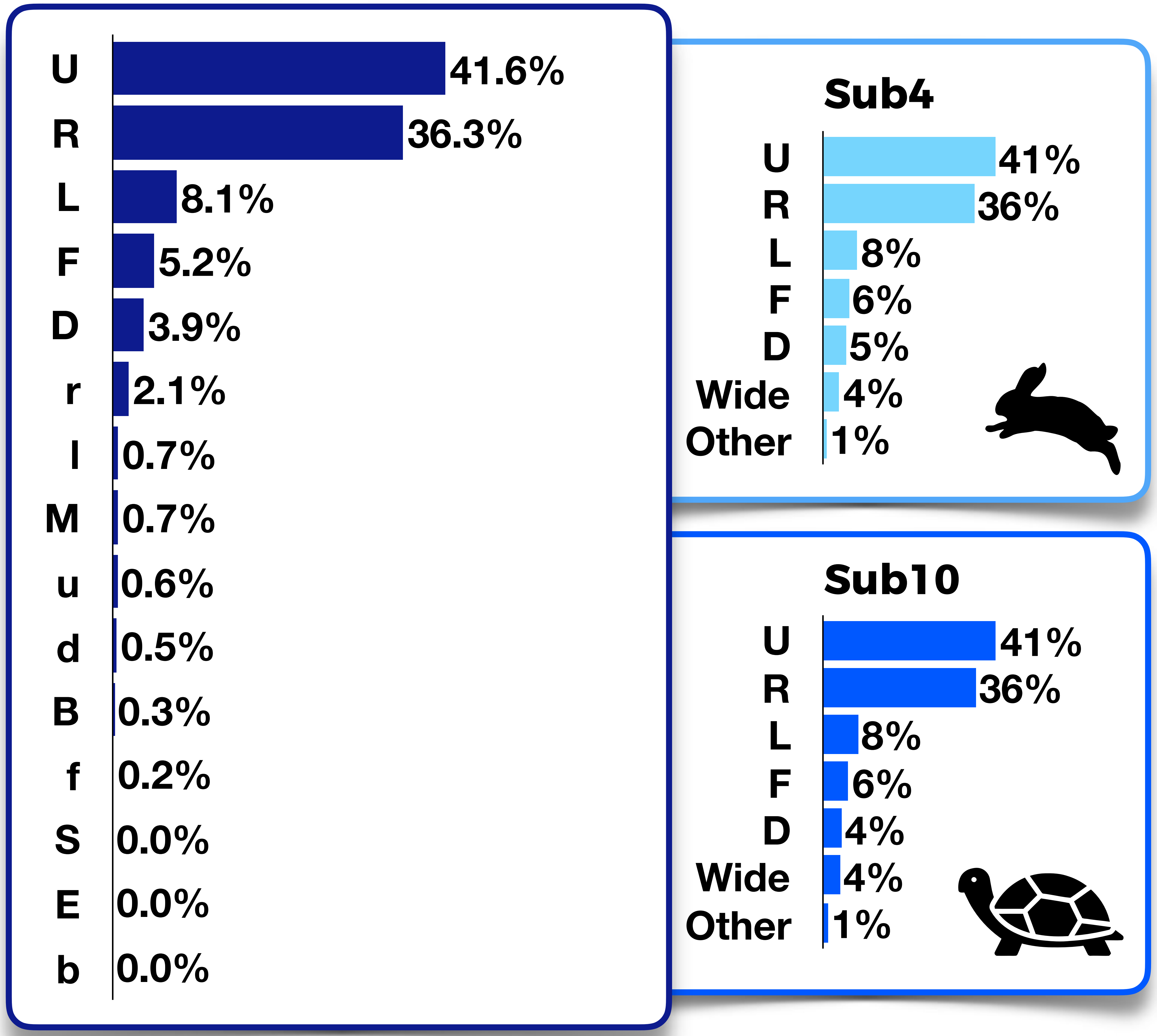
Planning is not muscle memory

Whilst cross can be planned during inspection, its execution is not a triggering of a memorised alg, as is the case of the following CFOP steps. Despite this, solvers are executing it at only 30% slower tps

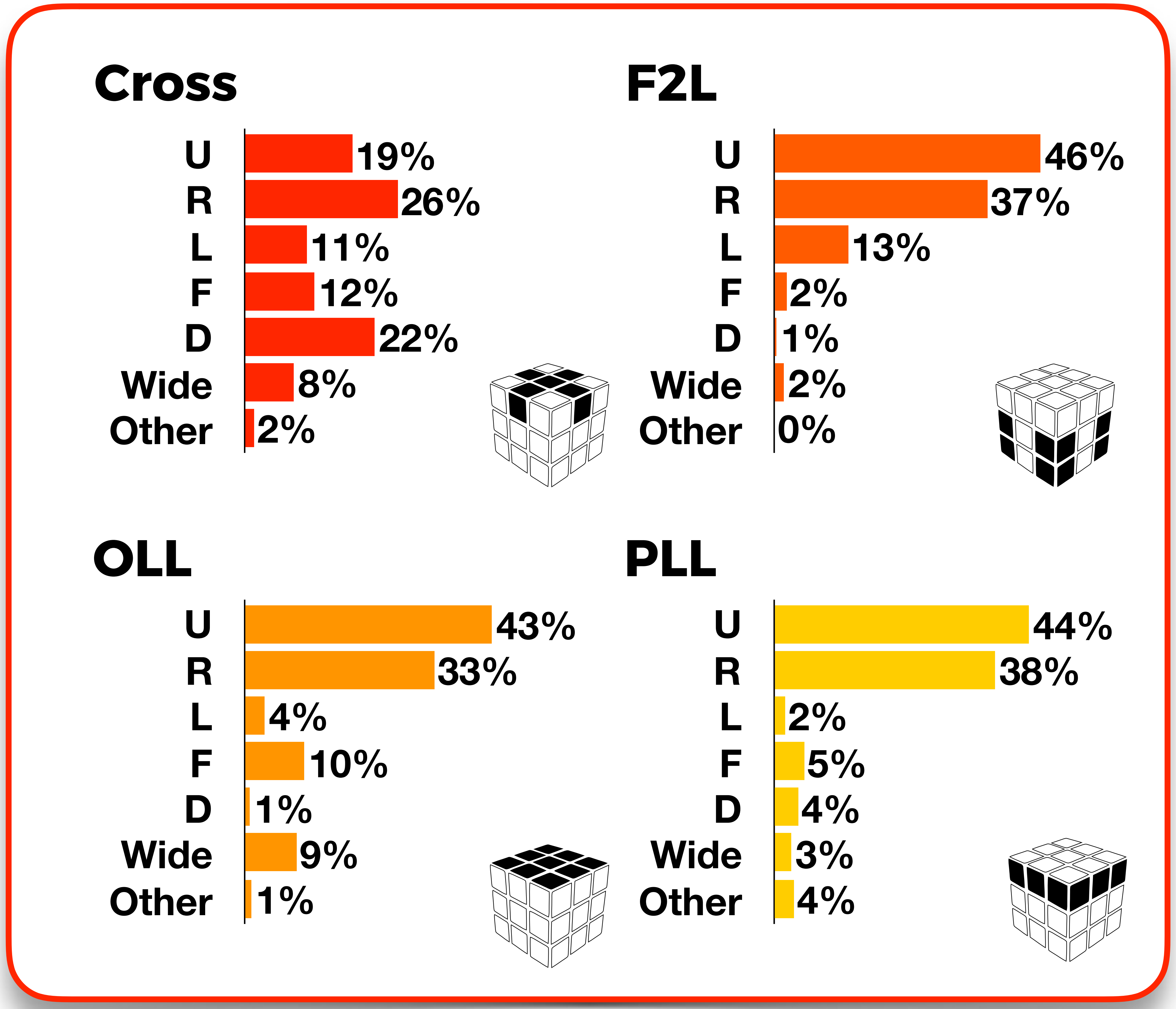
N=4000+

EVERYONE'S GOT MOVES, AND THEY ARE MOSTLY RU (SORRY S-SLICE CROWD!). CROSS IS THE MOST ECLECTIC STEP IN THE SOLVE

CFOP OVERALL MOVE USAGE



MOVE USAGE BY CFOP STEP



N=4000+

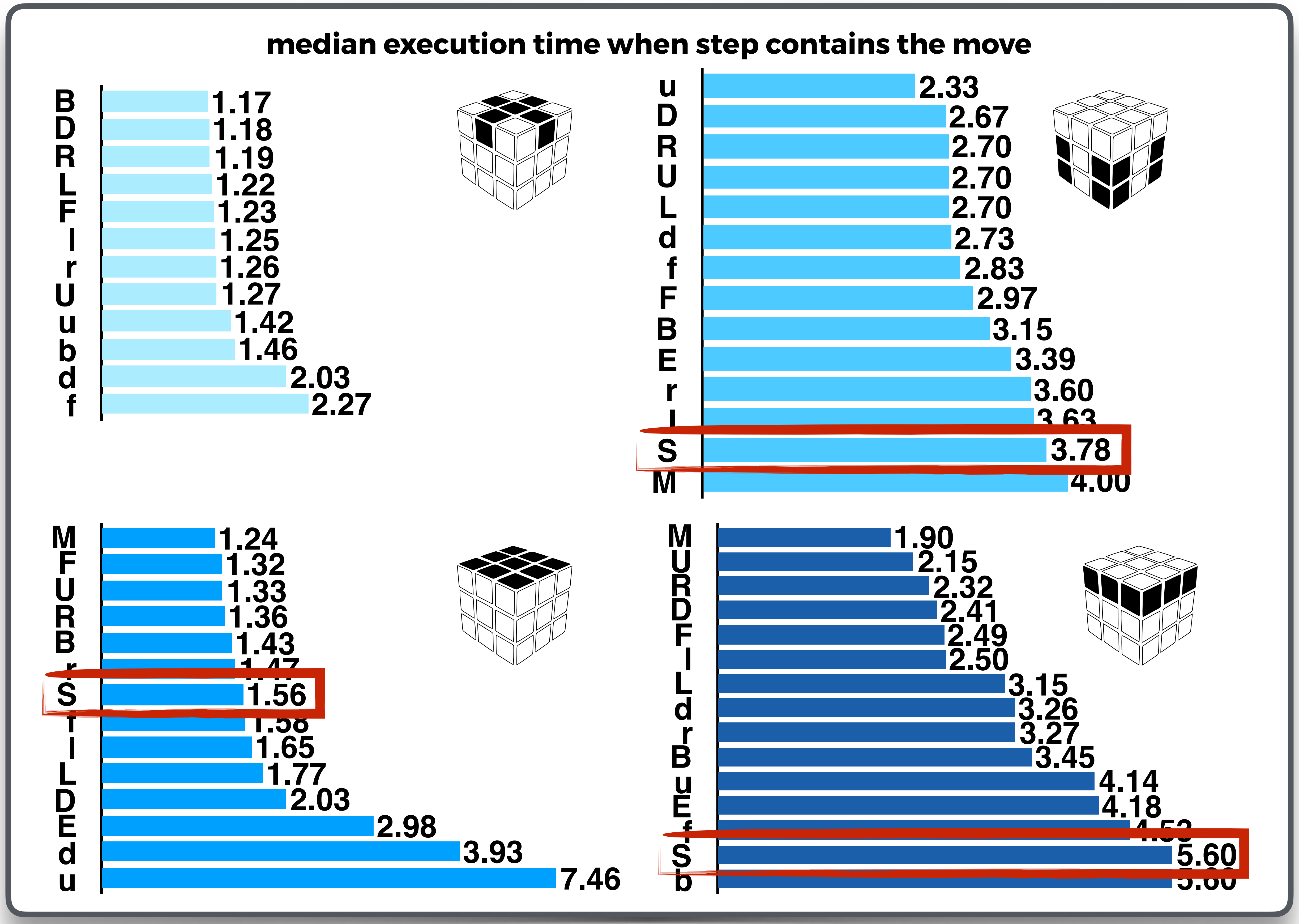
SOME MOVES ARE BETTER THAN OTHERS FOR DIFFERENT STEPS: S SLICES LOOK GREAT FOR OLL, LESS SO FOR ANYTHING ELSE

CFOP MOVE USAGE CORRELATION

If it's not one it will have to be the other

Unsurprisingly, there is a very strong negative correlation between Righty and Lefty moves, as well as (to a lower extent) R vs M moves: they serve similar purposes, but solvers who prefer one will use the others less

STEP EXECUTION TIME BY TYPE OF MOVE IT USES

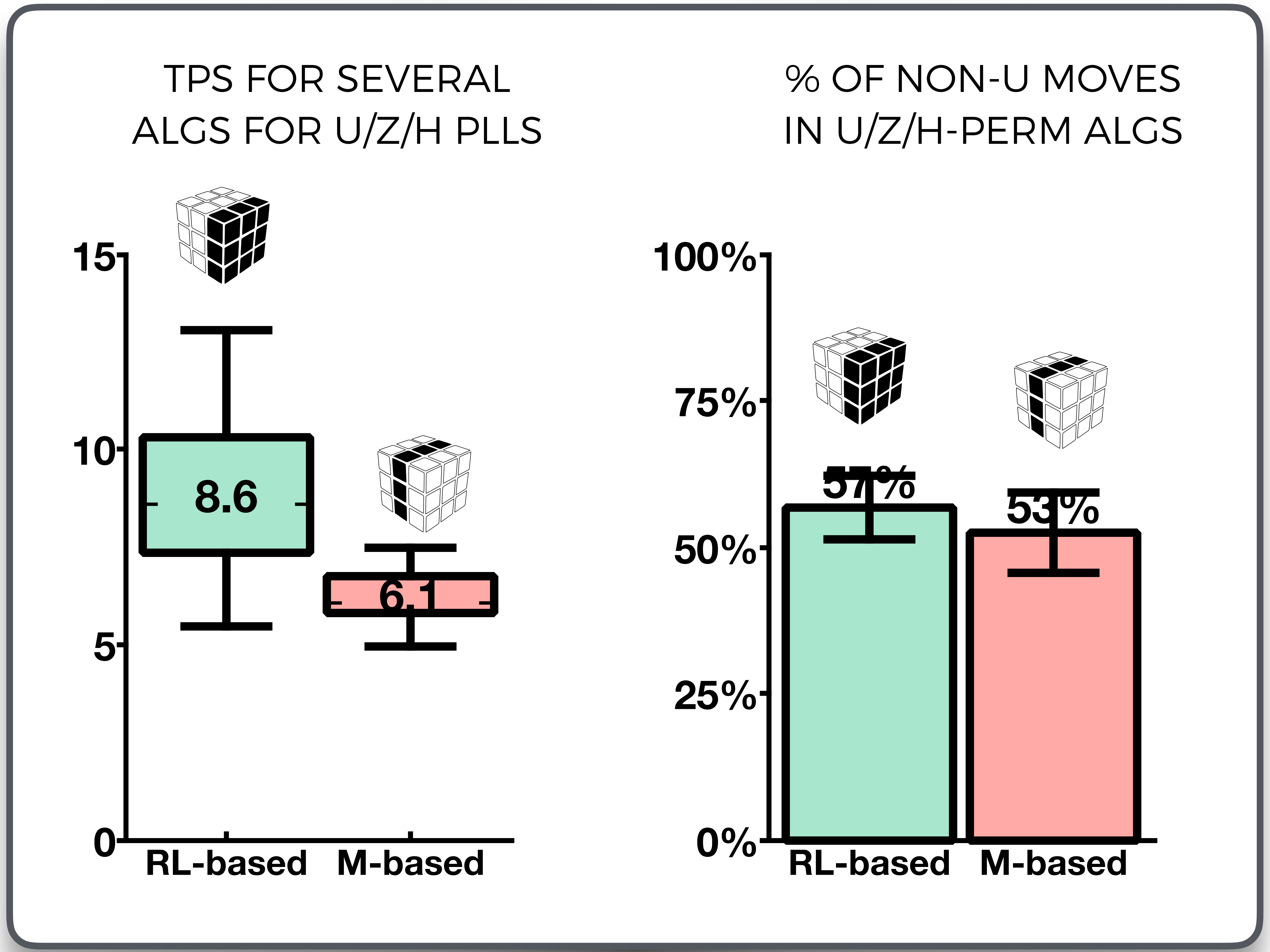


The Zoomer generation

Usage of S and E slices, as well as f move (e.g.) inserts are positively correlated. The recent hike in popularity of these moves seems to have brought all of them to the fore at the same time

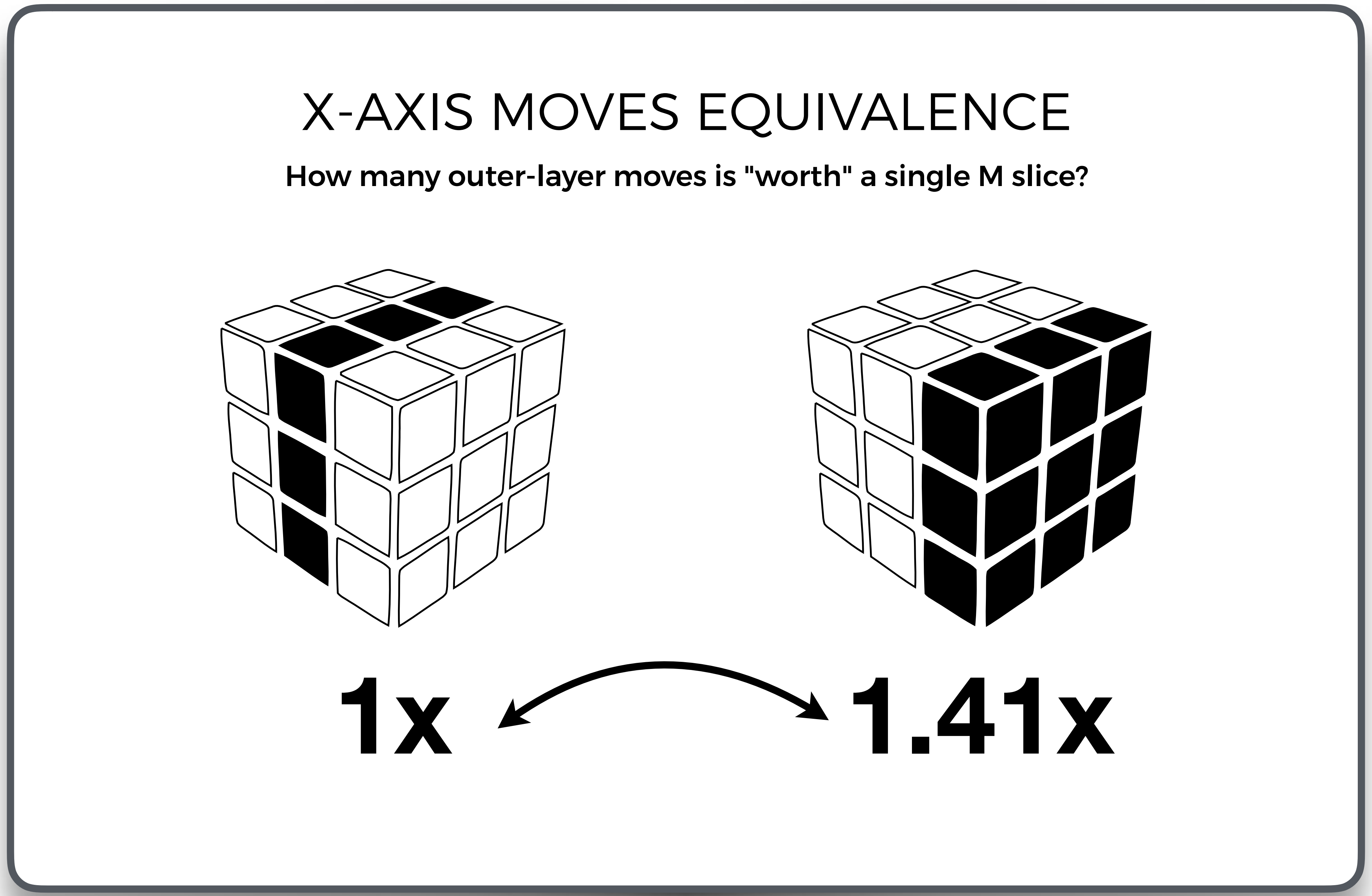
N=4000+

M MOVES : A BIT BETTER THAN ONE AND A HALF OUTER-LAYER TURNS



Why only these PLLs

By sticking to the same PLLs we control for alg recognition complexity, which would make a comparison of the simpler EPLLs and other PLLS unfair towards outer-layer-based algs



But what about Roux?

The current analysis utilises a strictly comparable regime of algs, where the "thinking" component has been taken out of the equation (same PLLs, just different algs). But what about Roux? We know that the tradeoff between lower-move count and lower-tps is present, but how much of it is due to Slice moves?

N=860

WHAT ABOUT WORLD-RECORD LEVEL SOLVES?

9 WR AND SUB-WR SOLVES AT A GLANCE

Exec. Time:
3.47 sec
or less

Rotations:
1.8 rot.
Incl. wide moves

Moves:
32.2 moves
on average

CROSS

Cross:	XCross:	XXCross:
56%	**22%**	**22%**

TPS:
9.7 turns
per second

TIME SPLITS

Cross:	F2L:	OLL:	PLL:
24%	**49%**	**21%**	**6%**

Gens:
5.6 gen

SKIPS

78% PLL skip

22% LL skip

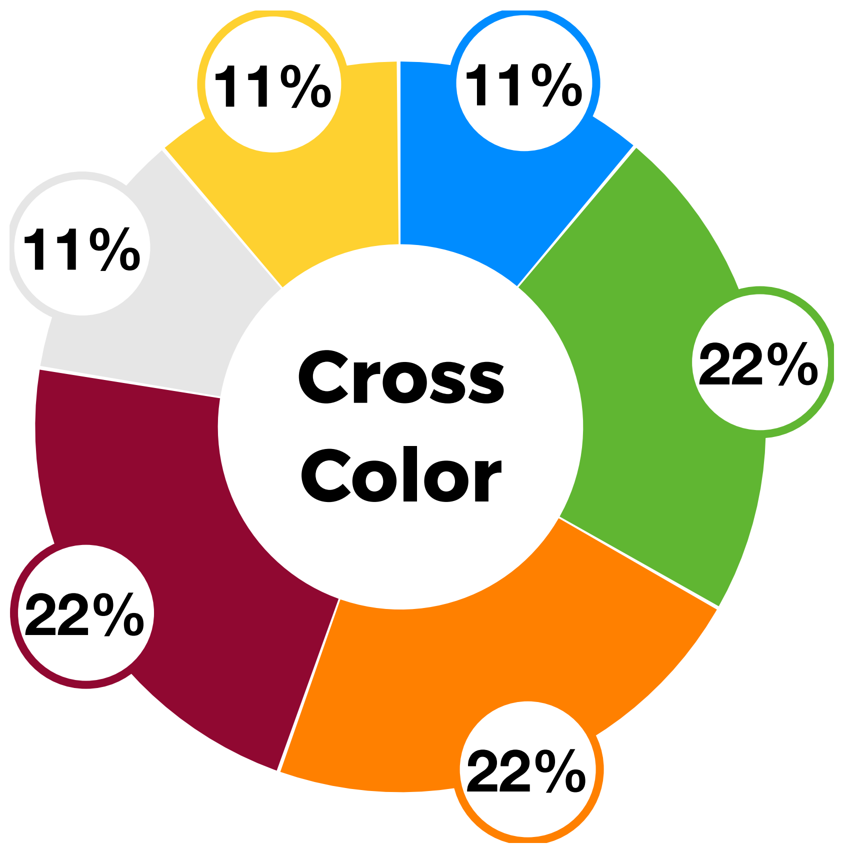
100% OLL(CP)

What is world-record level

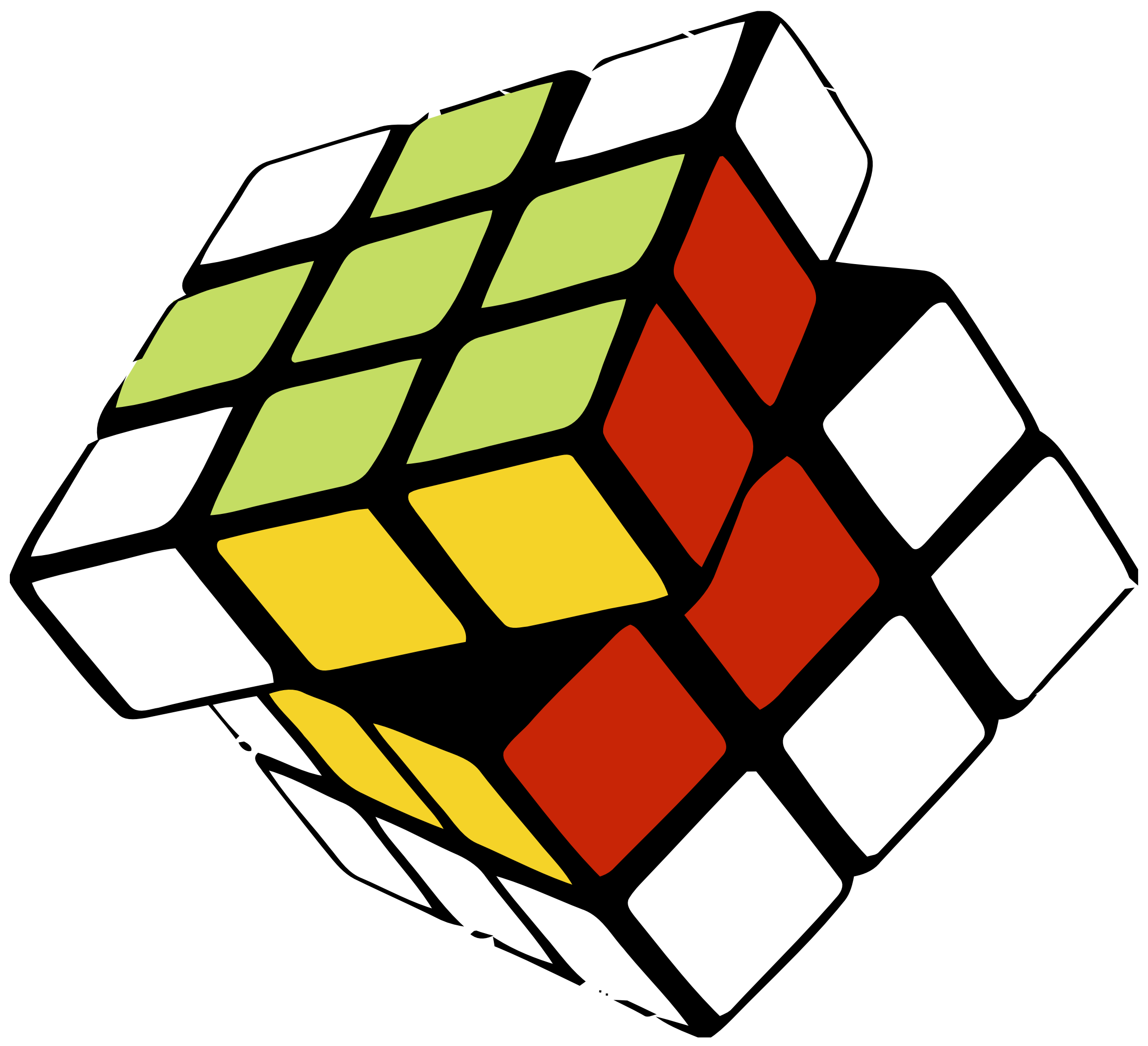
At the time of writing the current official 3x3 single WR stands at **3.47s by Yusheng Du**, for whom a number of solves (including his WR) are part of our data.

However, multiple solvers have managed to get faster solves in unofficial venues, some on cam, others reconstructed, some on stackmat, others on keyboard or smart cube.

Regardless, it is interesting to understand what it takes to get times as fast as the (current) world record



N=9



CHAPTER 2 : CROSS

A CROSS IS USUALLY DONE IN 6 MOVES, AT 5.2 TPS, WITH 1 ROTATION

Cross Stats

Exec. Time:
1.20 sec

Moves:
6.2 moves

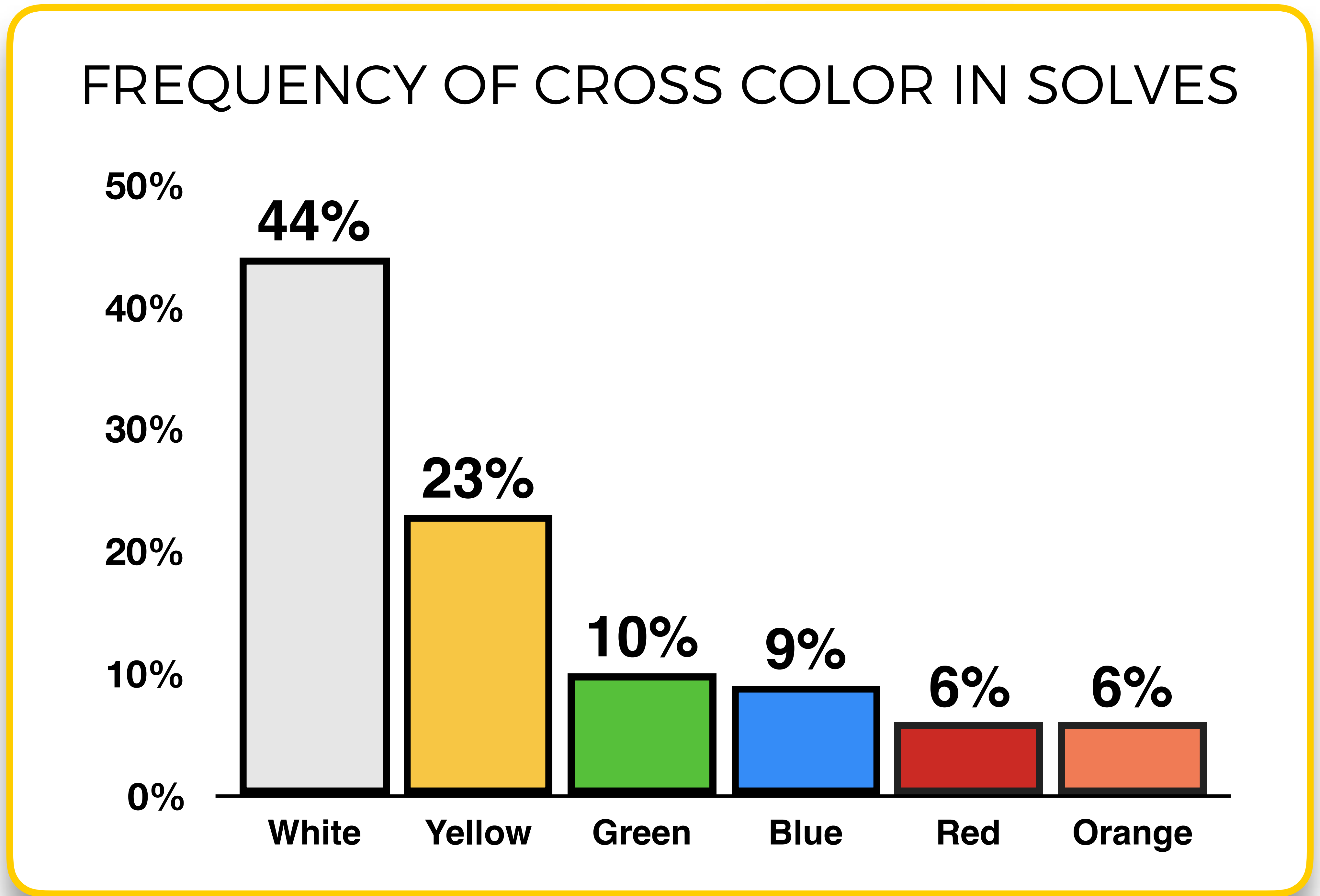
Gens:
4.1 gen

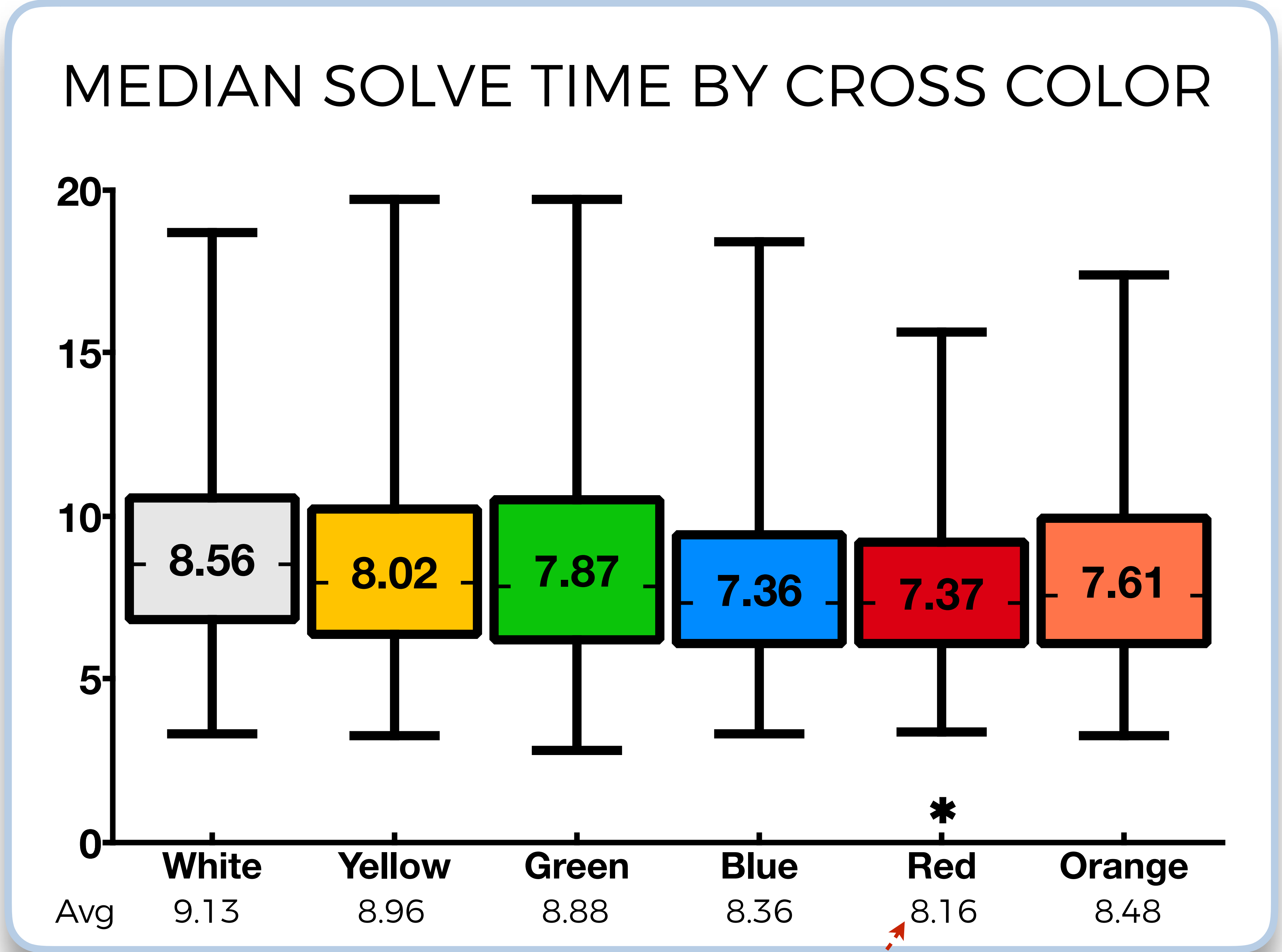
Rotations:
1.07 rot.
Incl. wide moves

SUB4 SOLVES	SUB 6 SOLVES	SUB 10 SOLVES
0.81 sec	**1.00** sec	**1.15** sec
6.0 moves	**6.8** moves	**6.5** moves
4.1 gen	**4.1** gen	**4.1** gen
0.79 rot.	**0.65** rot.	**0.80** rot.

N=4000+

CROSS COLOR : EVEN AT THESE LEVELS, WHITE AND YELLOW COME OUT ON TOP (BY A FACTOR OF 2 EACH RESPECTIVELY)





or not seeing it enough?

Or is it just the least frequent color, and therefore we don't have enough bad solves?

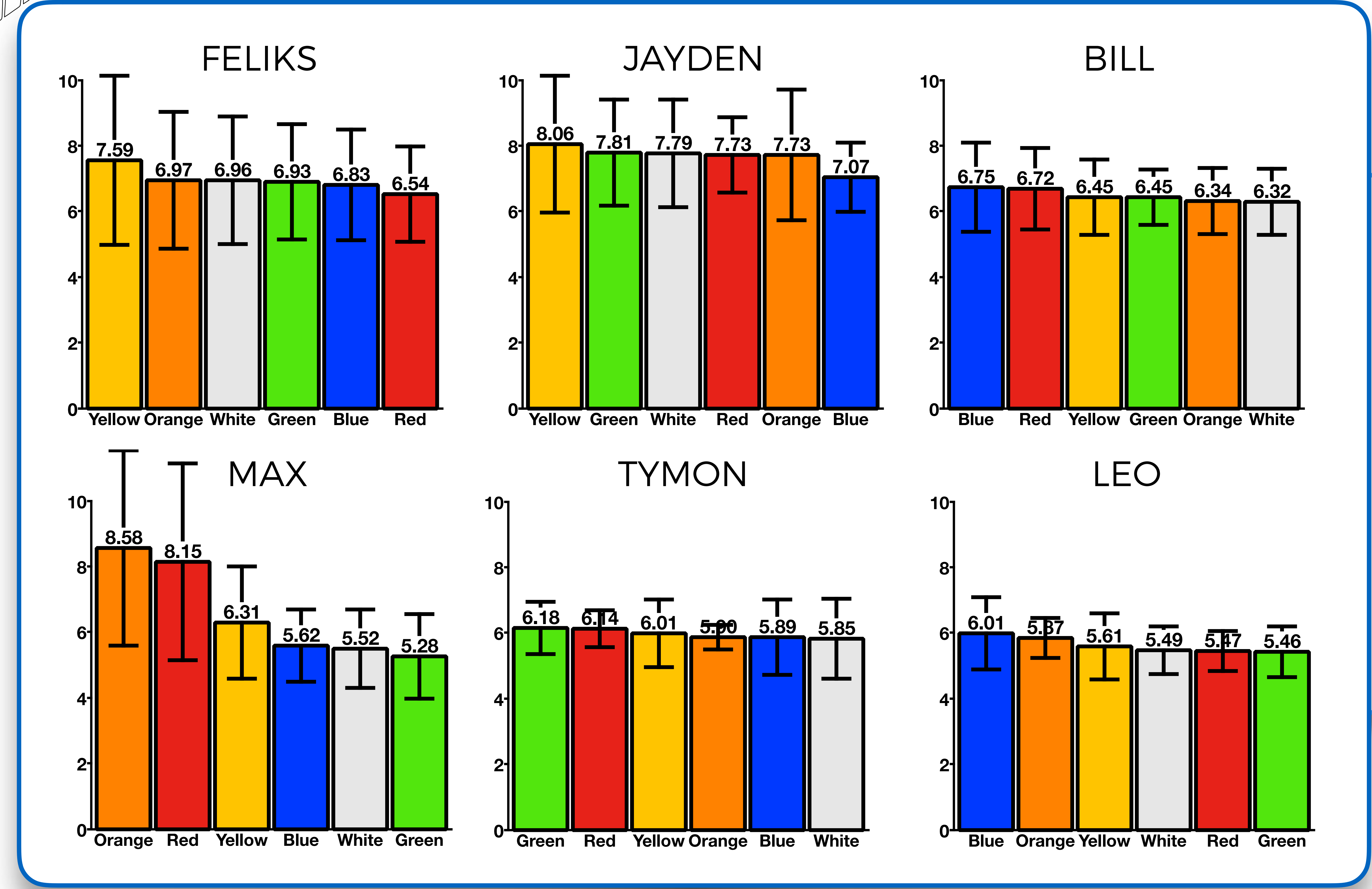
Seeing red

Is there something intrinsically "better" with red cross that makes it slightly faster than the other colors, or is that a fluke? Might physiological adaptations to contrast recognition relating to red be at play here? We need to dig deeper.

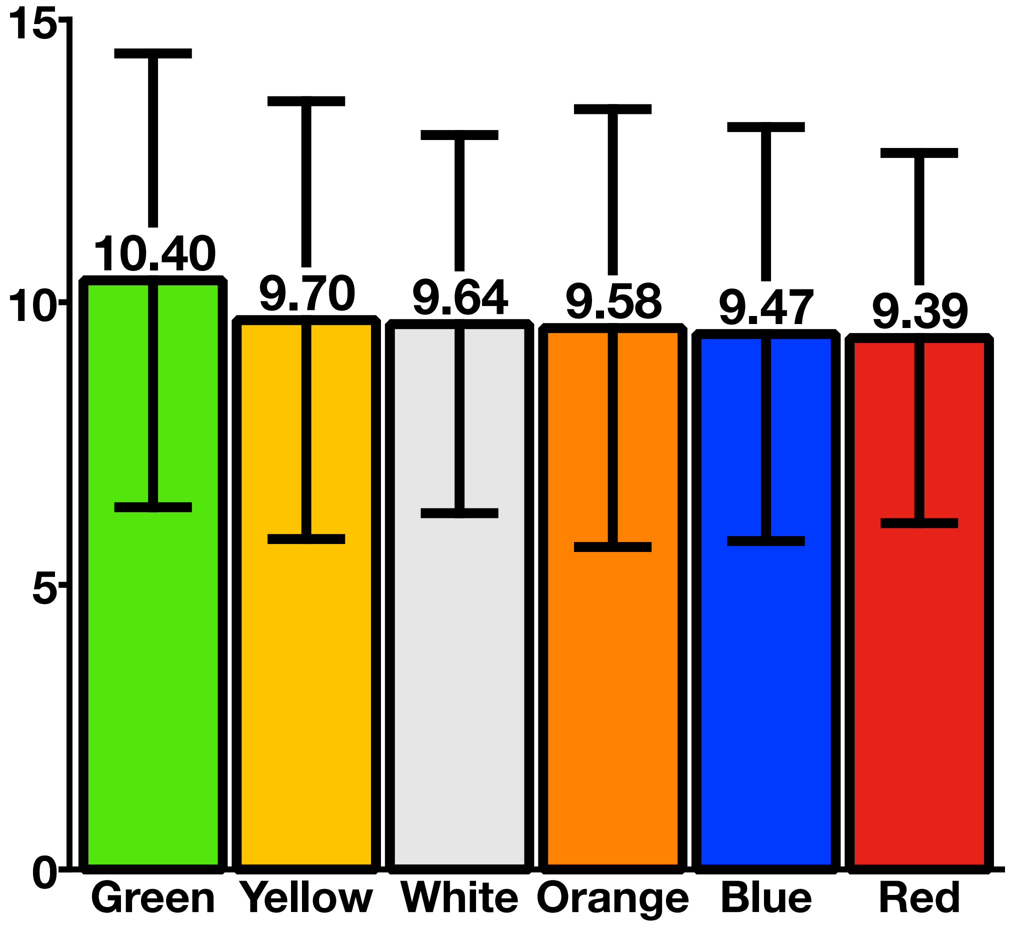
N=4000+

BUT WHEN WE LOOK ON A SOLVER BY SOLVER BASIS, THE STORY CHANGES

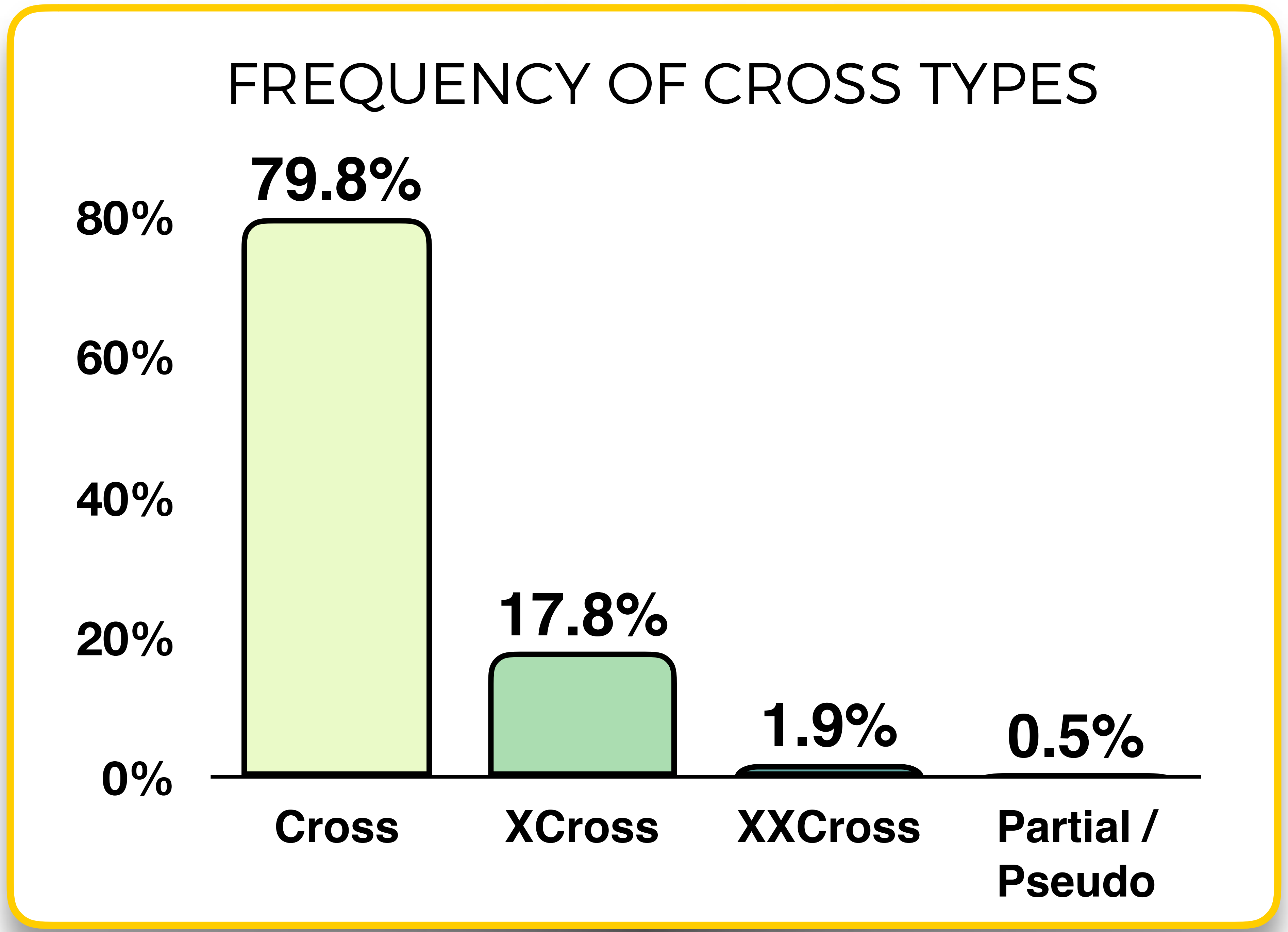
AVERAGE SOLVE TIME BY CROSS COLOR

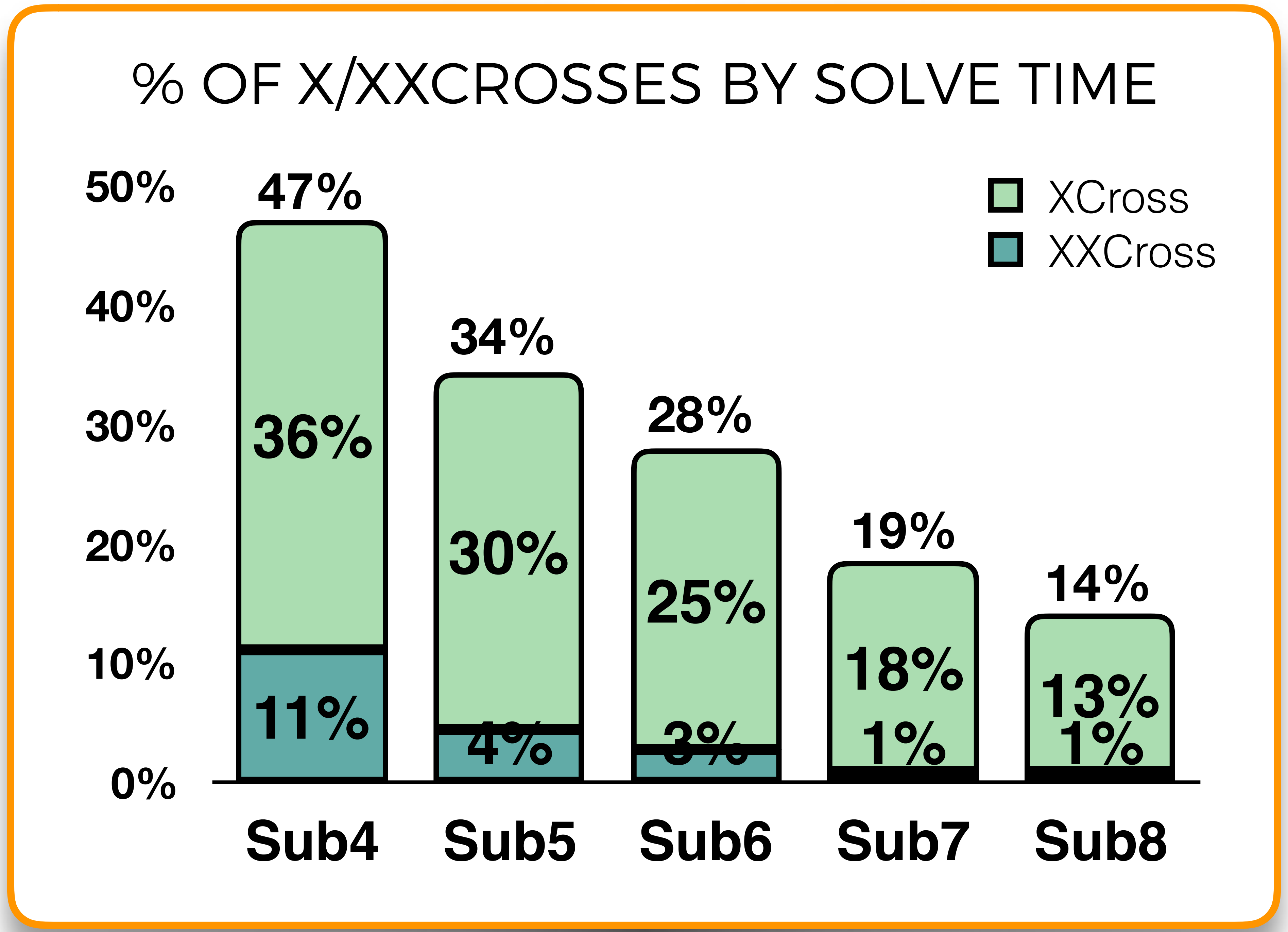


AVERAGE OF ALL SOLVERS EXCEPT THOSE FOR WHOM WE HAVE 80+ SOLVES



X & XX CROSSES ARE RELATIVELY FREQUENT IN GENERAL (ALMOST 1/5 SOLVES OVERALL), BUT THEY BECOME CRUCIAL FOR THE FASTEST SOLVES; PSEUDO AND PARTIAL SEEM TO BE MORE NICHE AND NOT AS ESSENTIAL



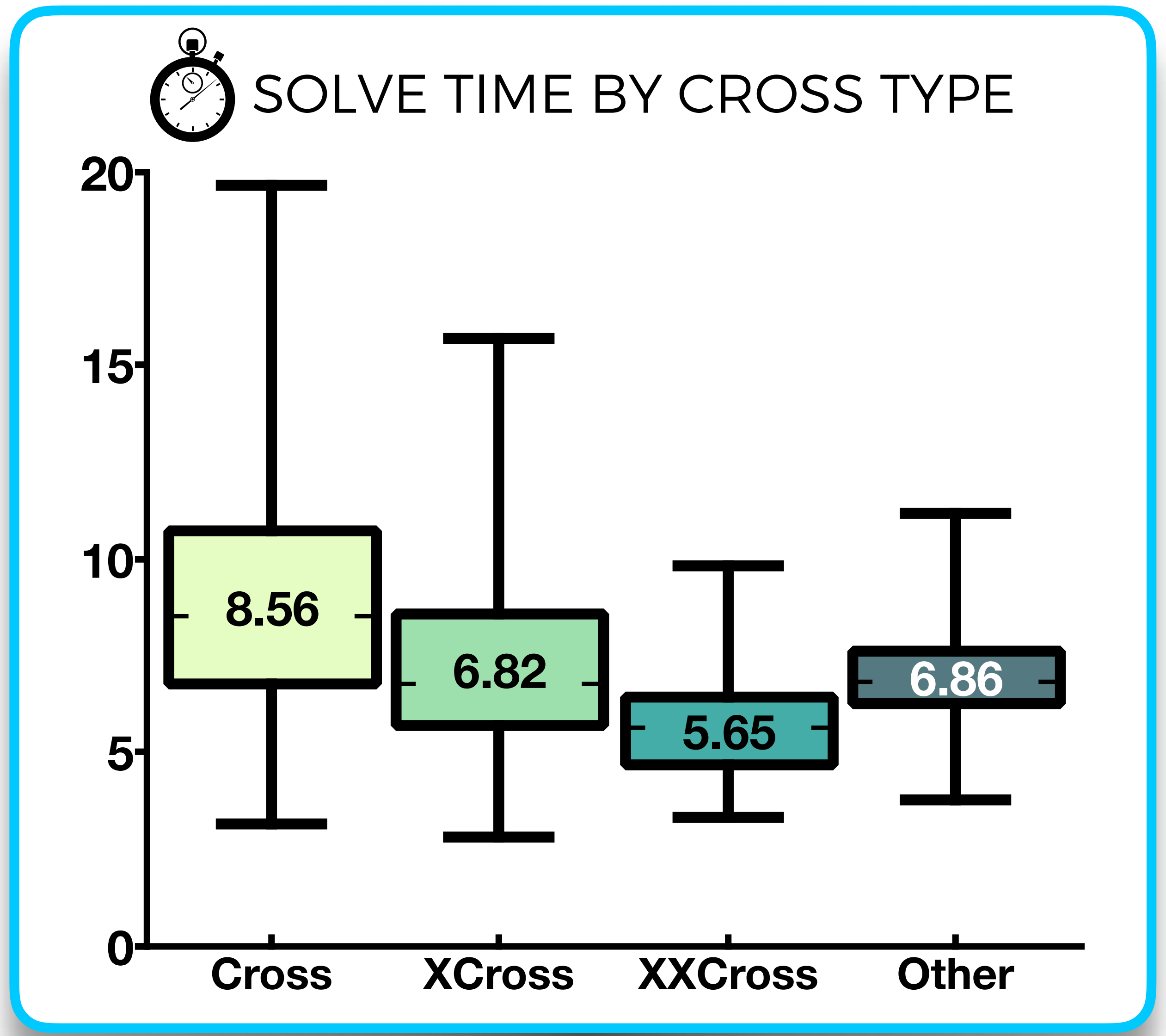


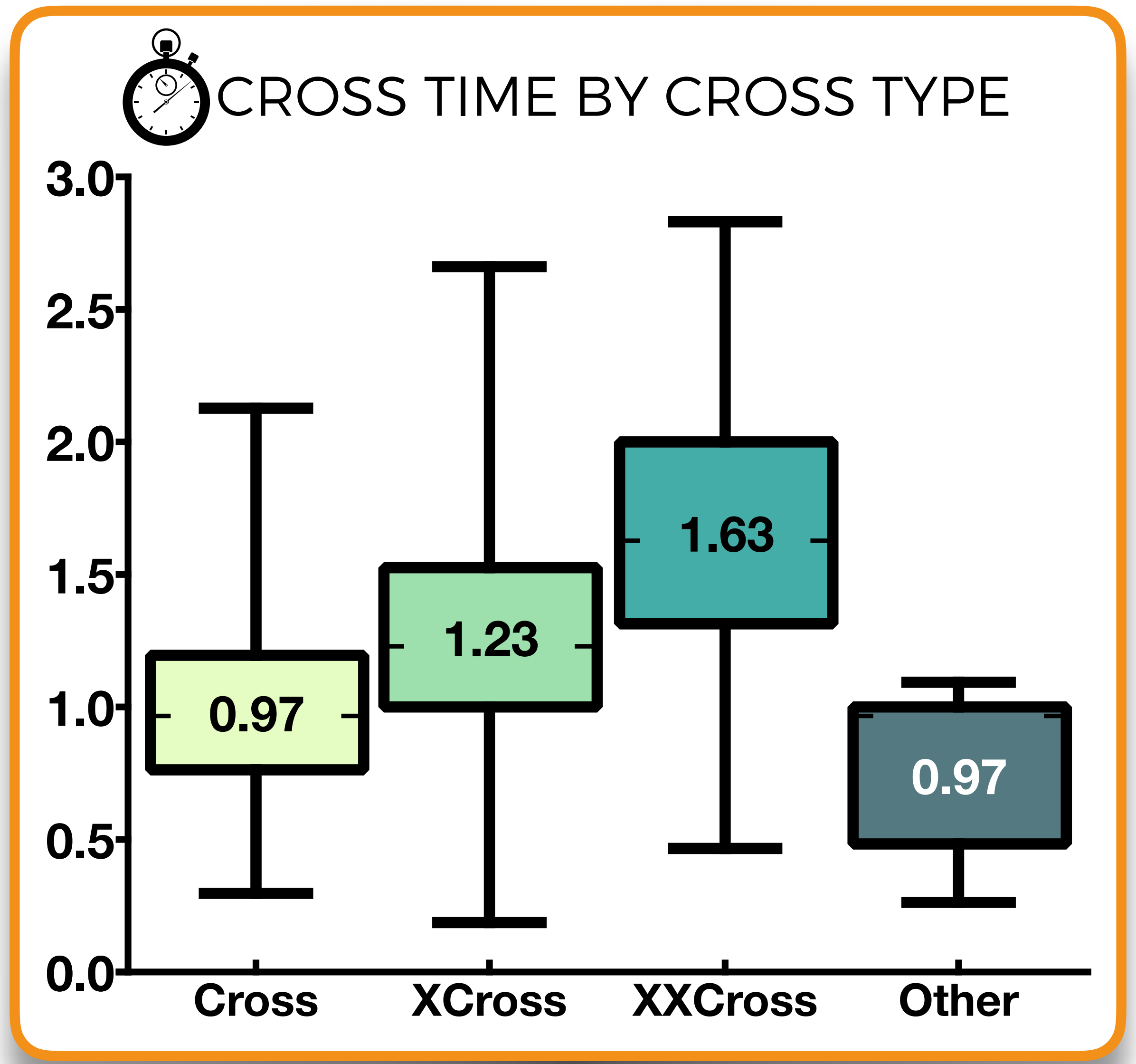
Everything starts with a good start

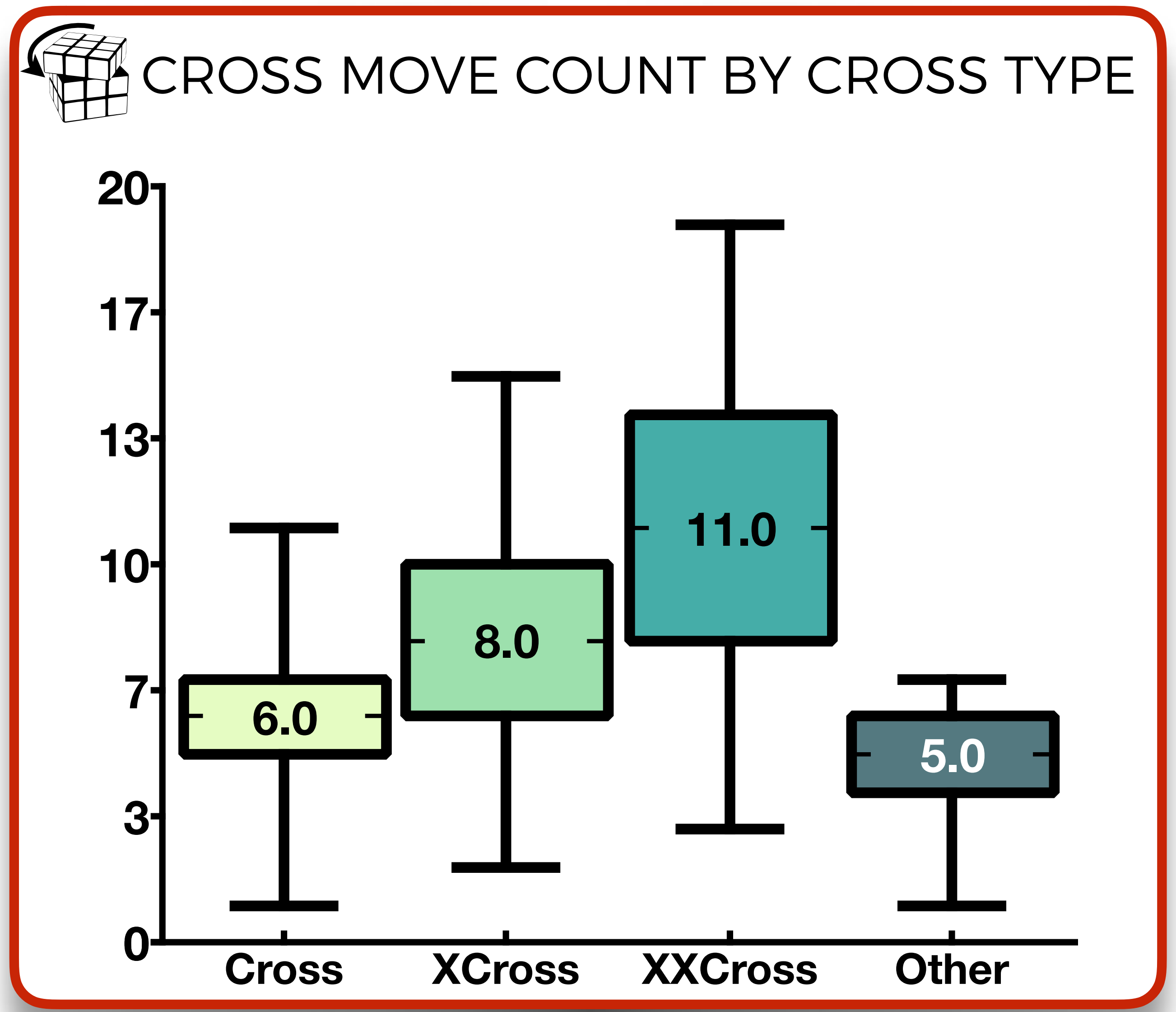
The faster the solve, the more likely it started out with a complex (and efficient) X(X)cross solution. Maybe it's not a required condition, but it looks like something worth working towards

N=4000+

X & XX CROSSES ARE RELATIVELY FREQUENT IN GENERAL (ALMOST 1/4 SOLVES OVERALL), BUT THEY BECOME CRUCIAL FOR THE FASTEST SOLVES; PSEUDO AND PARTIAL SEEM TO BE MORE NICHE AND NOT AS ESSENTIAL







Tradeoffs

XXCross can shave more than 0.5 seconds on a solve, this explains why they appear so often in good solves

MEDIAN SOLVE TIME FOR F2L	
1st pair	**1st+2nd pair**
0.63s	**1.40s**

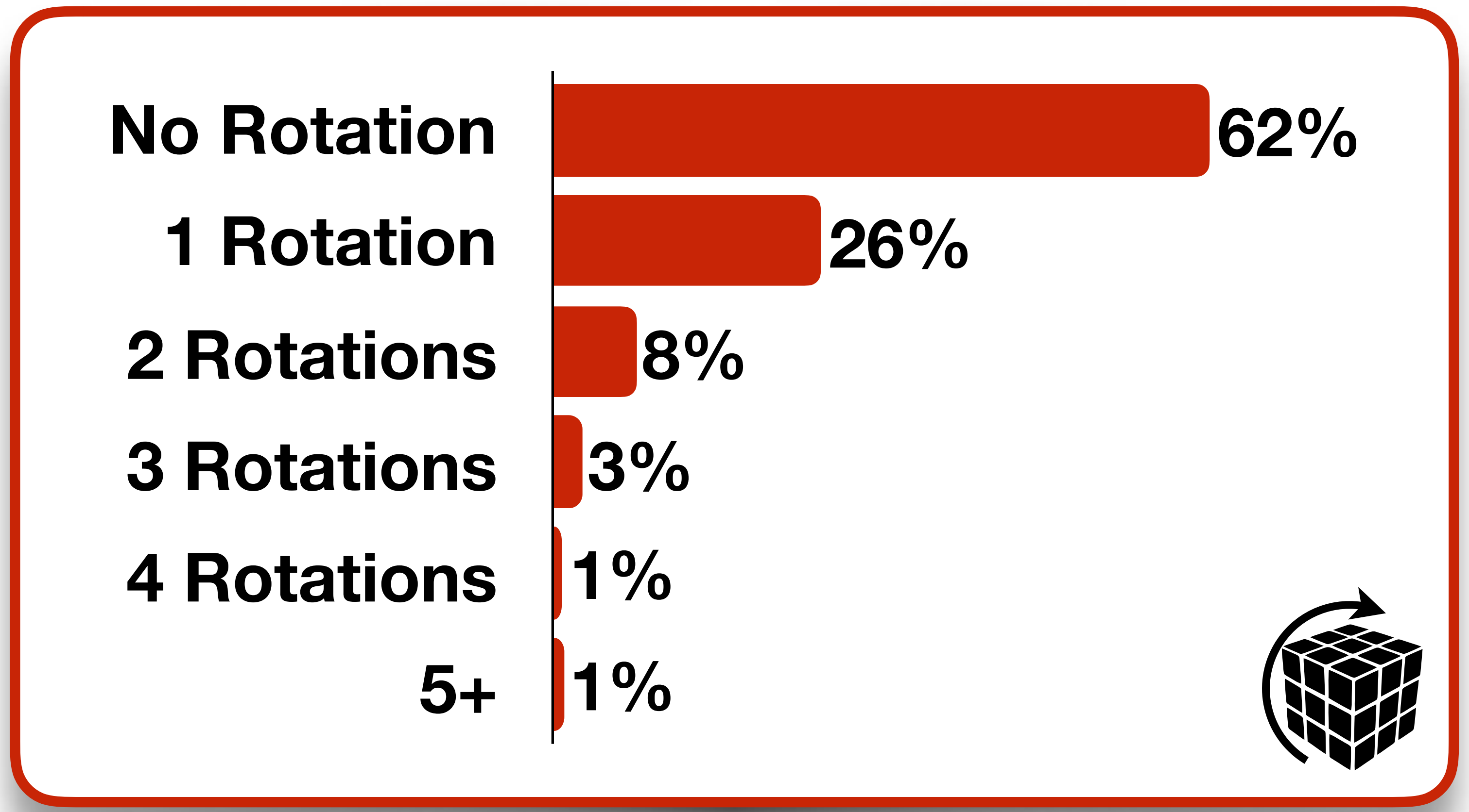
→

ABSOLUTE GAIN FROM X/XXCROSS	
XCross	**XXCross**
0.31s	**0.60s**

N=4000+

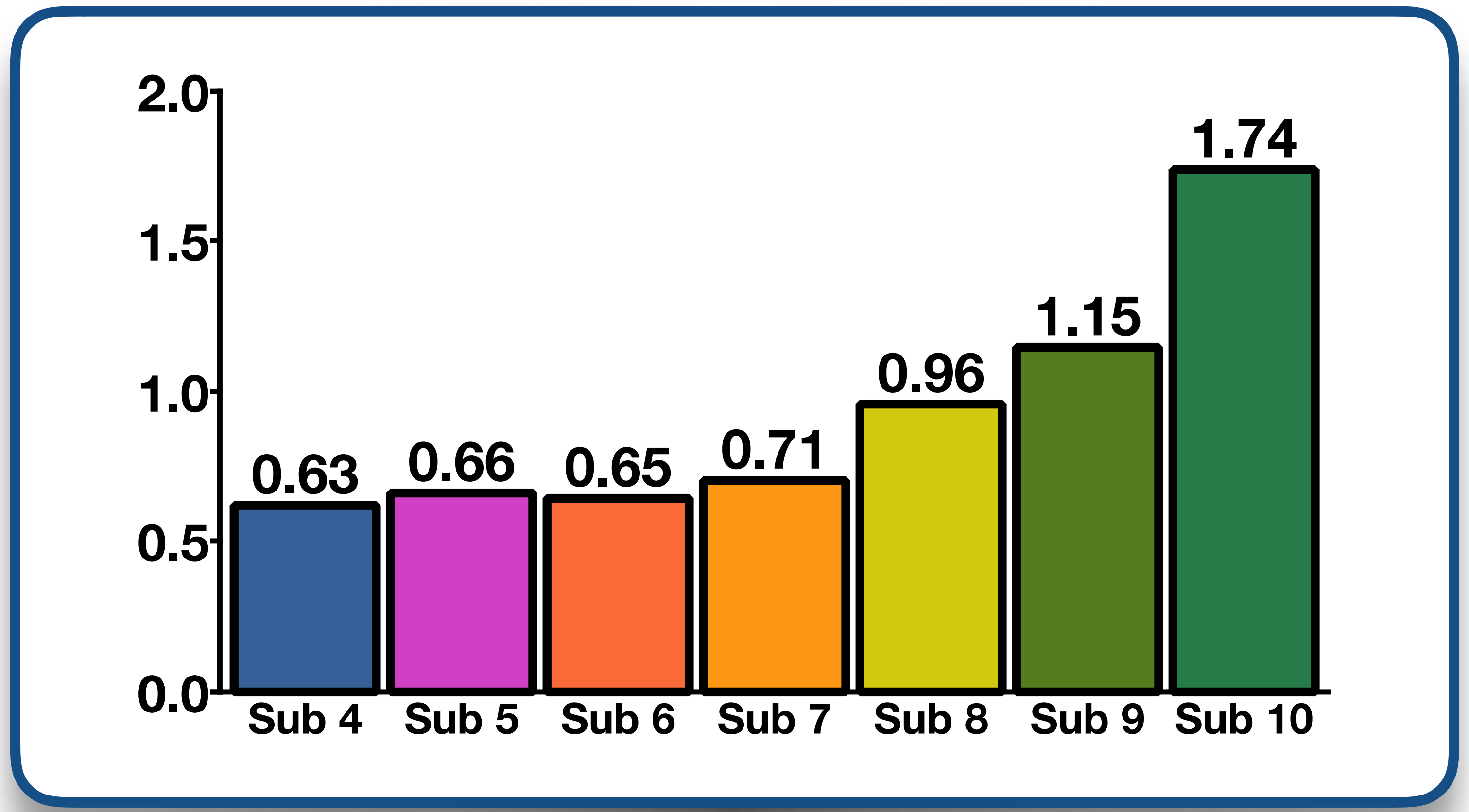
1/3 OF SOLVES HAVE NO ROTATIONS OR WIDE ROTATIONS IN CROSS, BUT THE FASTEST SOLVES HAVE FEW

OF ROTATIONS DURING CROSS

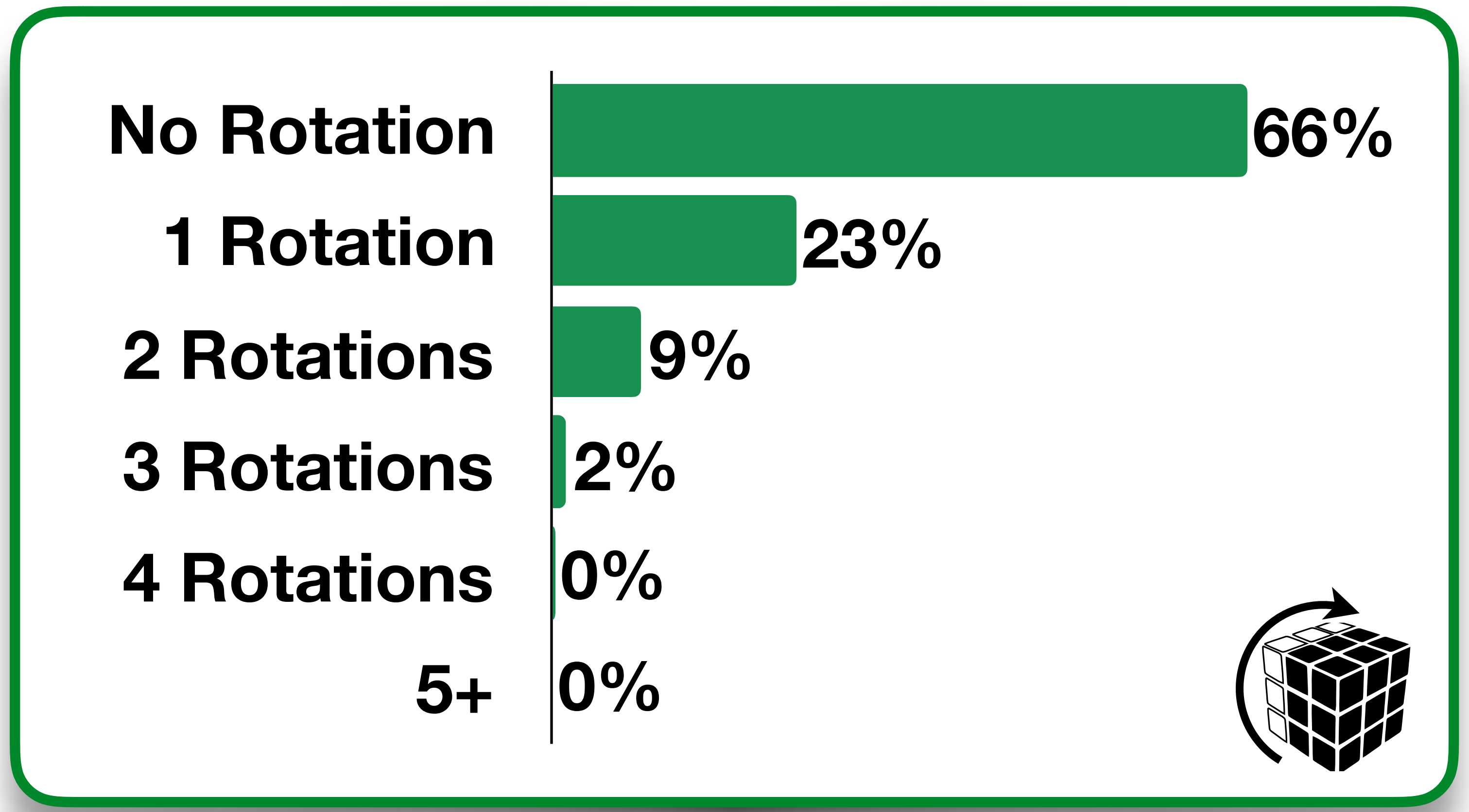


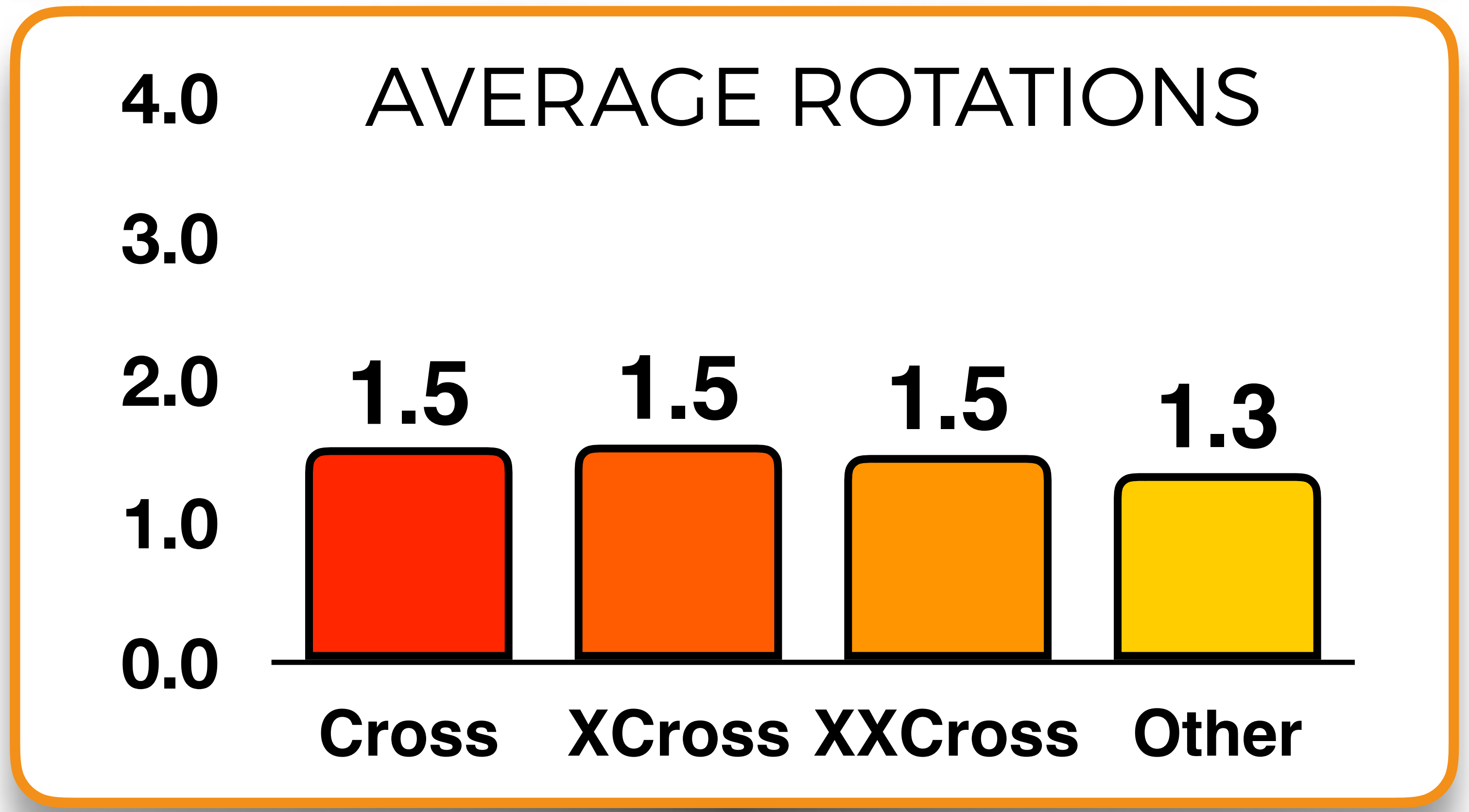
TOTAL ROTATIONS BY SOLVE TIME

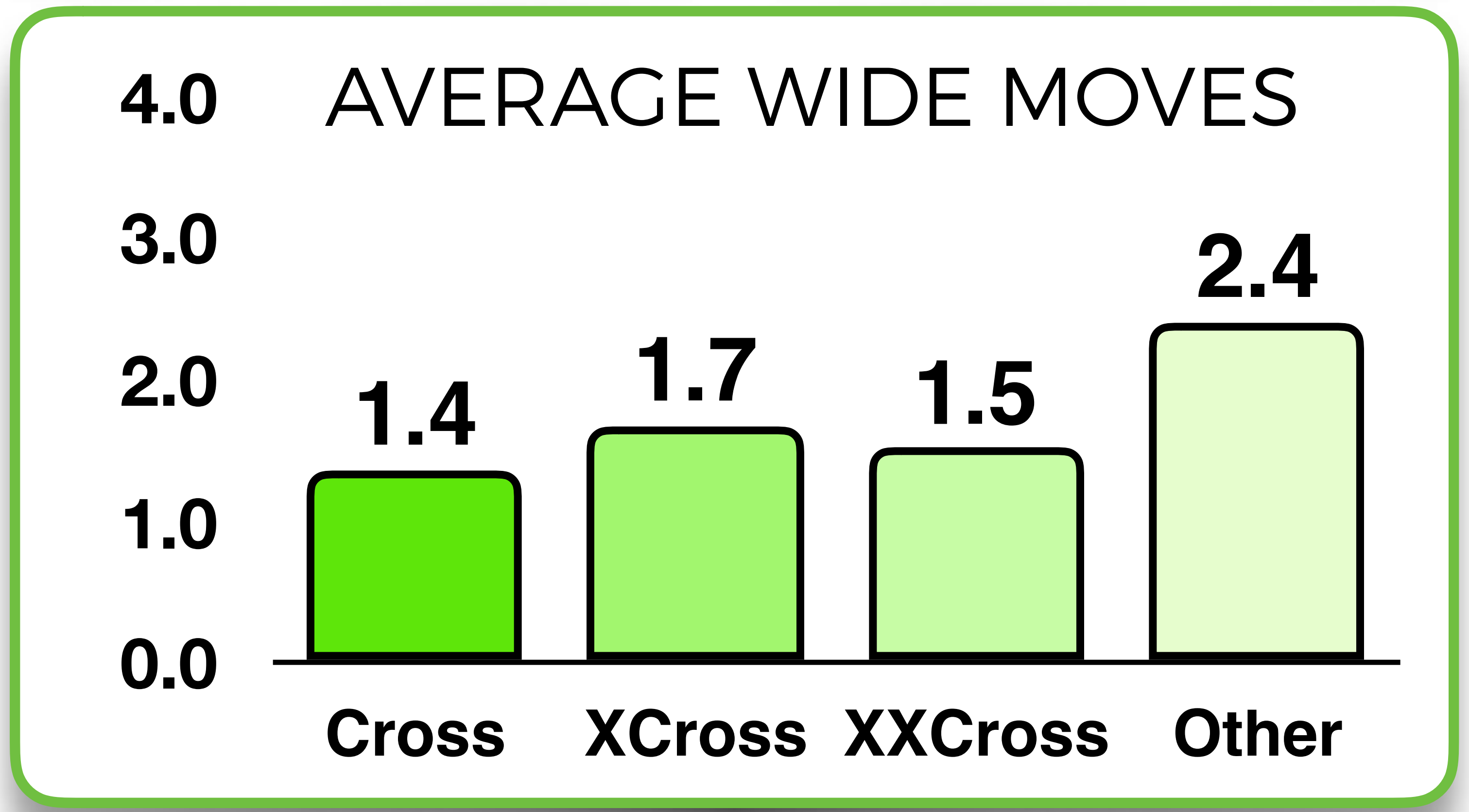
Rotations + wide moves



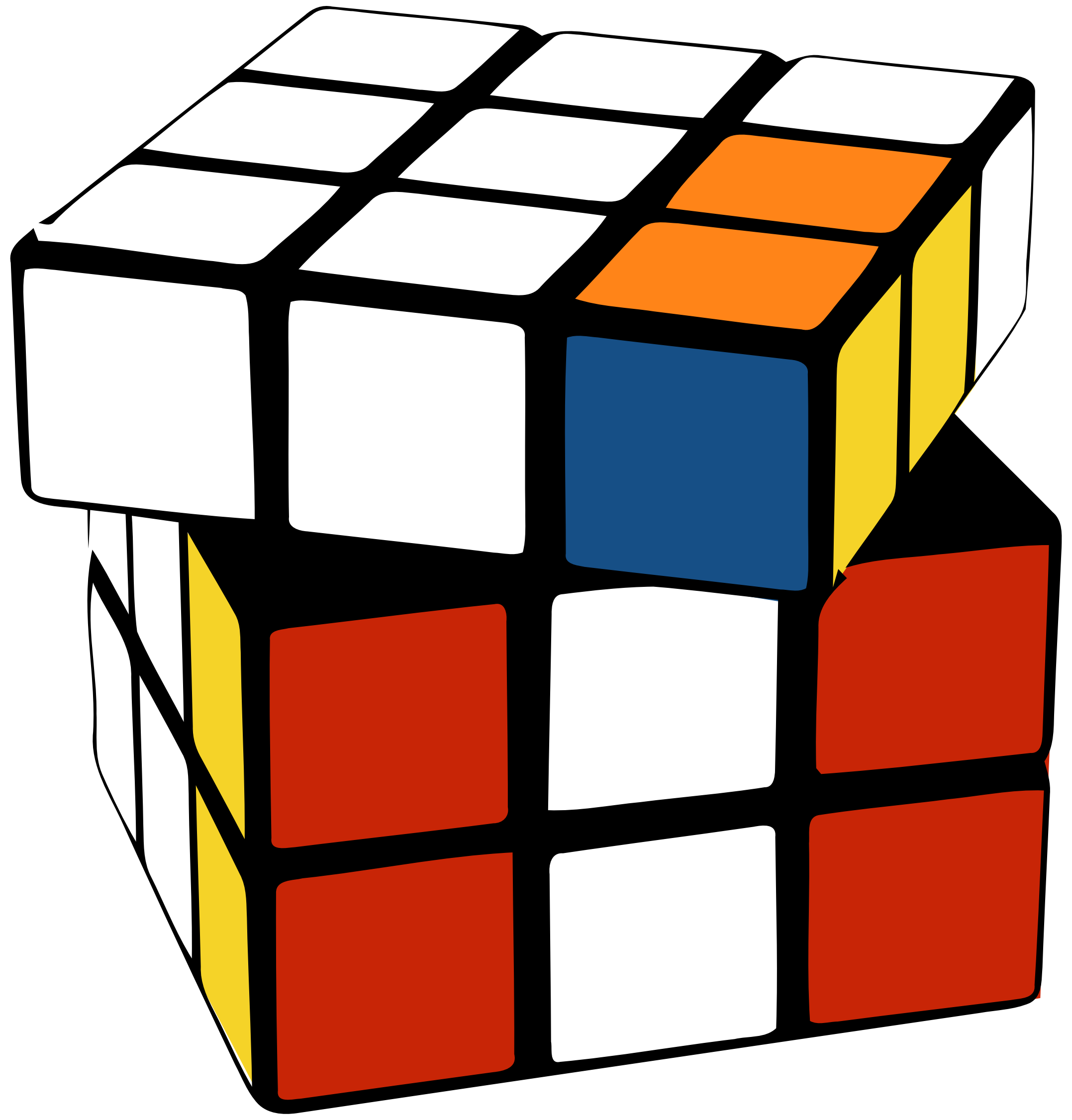
OF WIDE MOVES DURING CROSS







N=4000+



CHAPTER 3 : FIRST 2 LAYERS

F2L PAIRS TAKE 8 MOVES ON AVERAGE, BUT TO GO FASTER THIS NEEDS TO GO DOWN (AND THE SOLVE NEEDS TO LET YOU DO IT!)

Per f2l Pair

Exec. Time: **0.85** sec

Moves: **8.0** moves

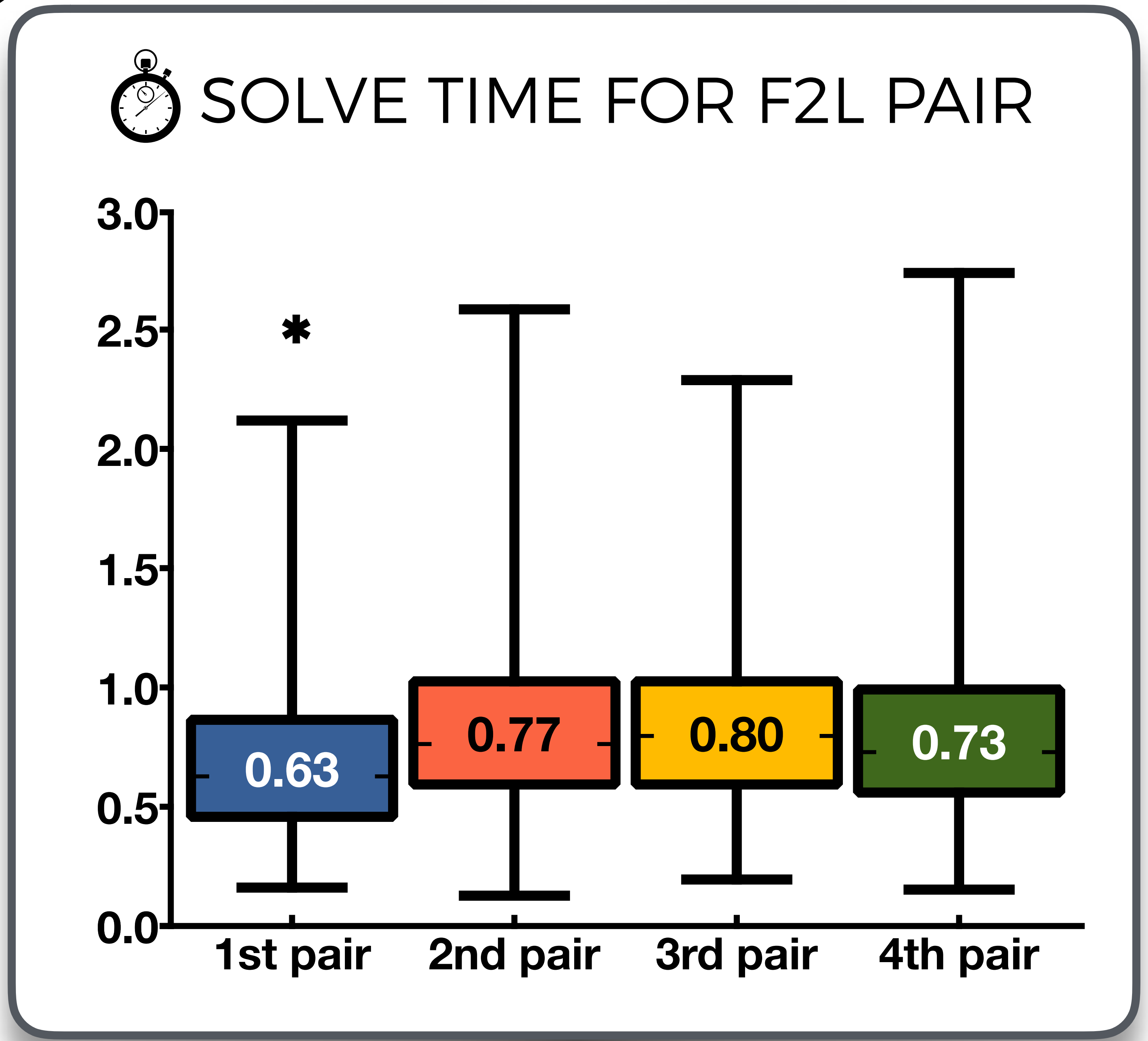
Gens: **2.4** gen

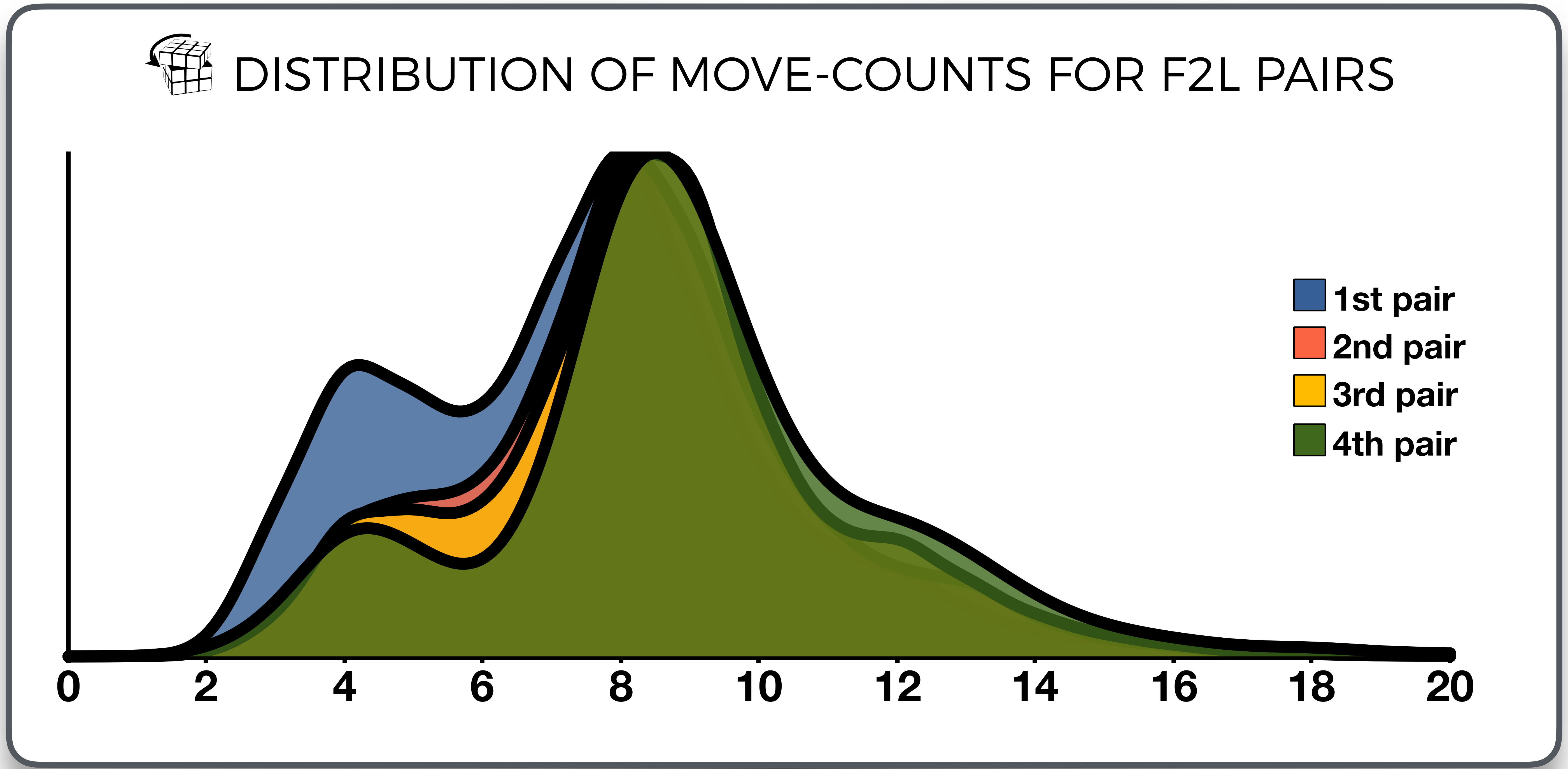
Rotations: **0.76** rot.
Incl. wide moves

SUB4 SOLVES	SUB 6 SOLVES	SUB 10 SOLVES
0.59 sec	**0.67** sec	**0.80** sec
5.9 moves	**6.7** moves	**7.6** moves
2.4 gen	**2.4** gen	**2.3** gen
0.47 rot.	**0.53** rot.	**0.67** rot.

N=4000+

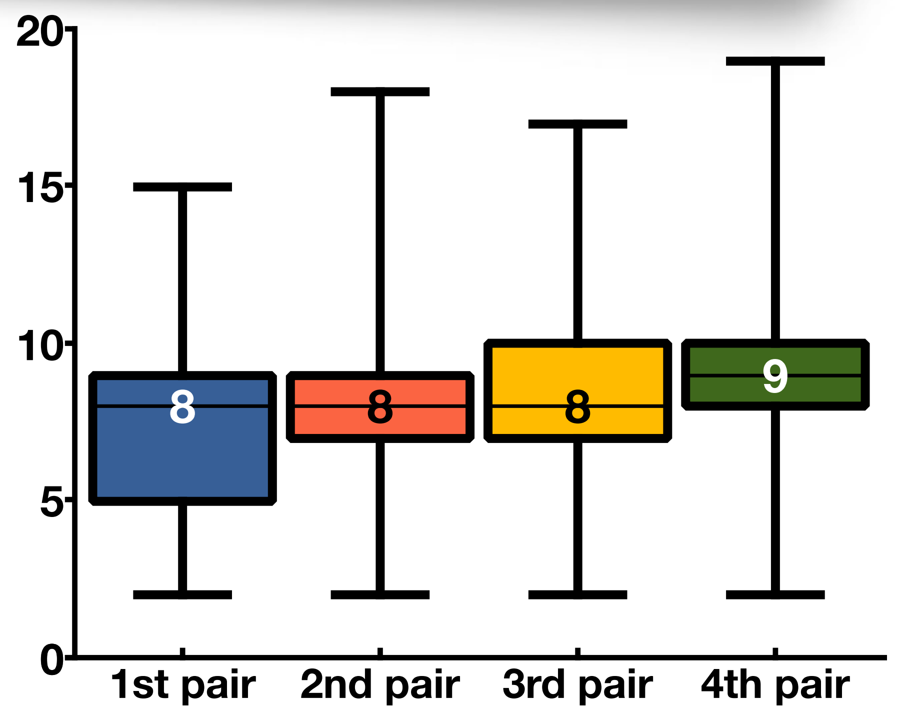
FIRST PAIR TENDS TO BE FASTER (THE POWER OF CROSS +1), THE OTHER PAIRS ARE VERY COMPARABLE; IN TERMS OF MOVE-COUNT, 8 IS THE GENERAL RULE





Cross+1 and planning | First pair ends up being 18% faster than the other pairs (on average), the effect of inspection, or the choice of "easy pickings" at the beginning of the solve

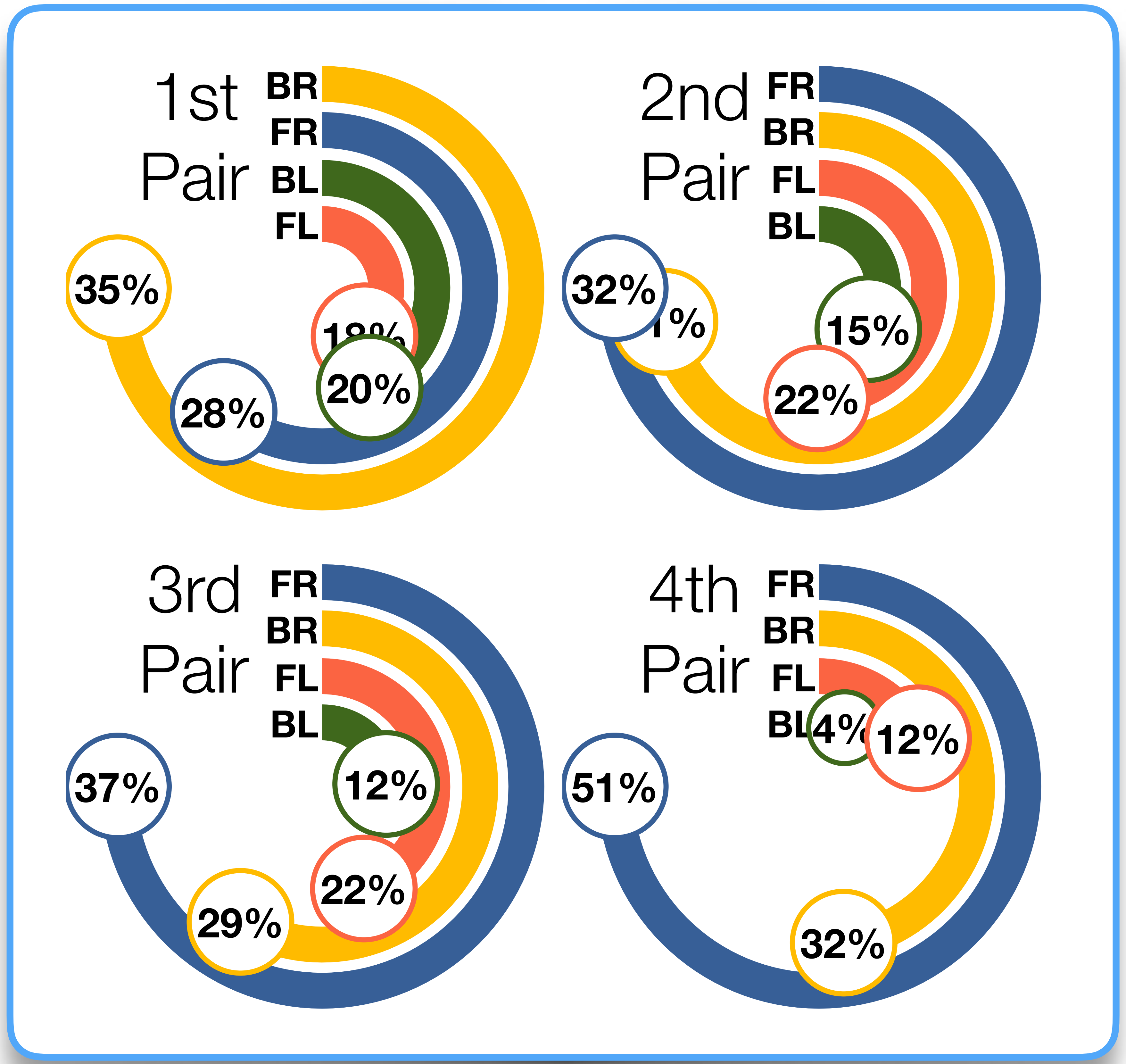
A case of good cases | The typical move count is basically the same for all pairs (8), but the advantage of picking an "obvious pair" shows the higher occurrences of short first pairs



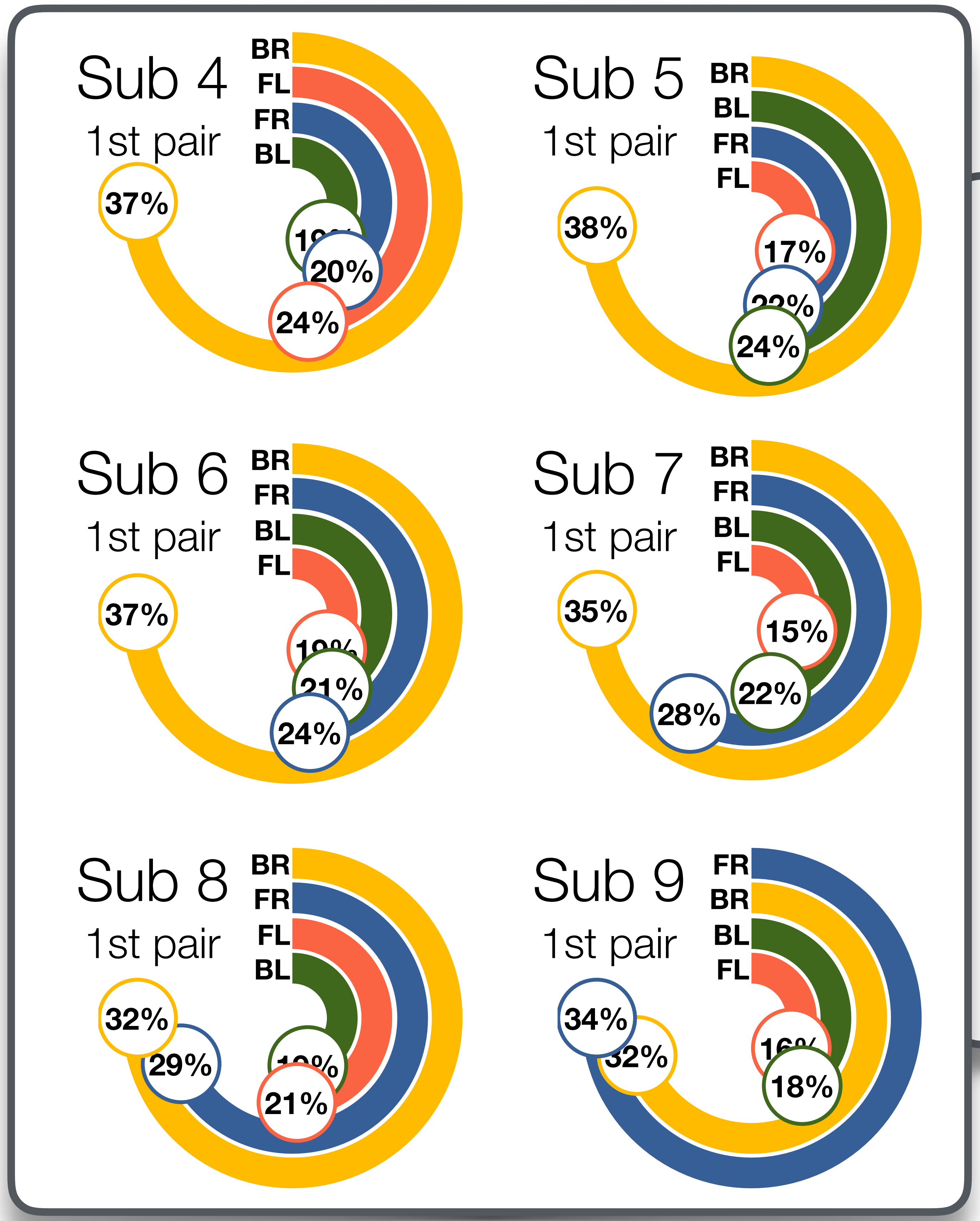
N=4000+

1ST SLOT ENDS UP IN THE BACK-RIGHT THE MOST OFTEN, AND THE FASTER THE SOLVE, THE MORE LIKELY IT STARTS THERE. LAST SLOT ENDS FRONT-RIGHT HALF OF THE TIME

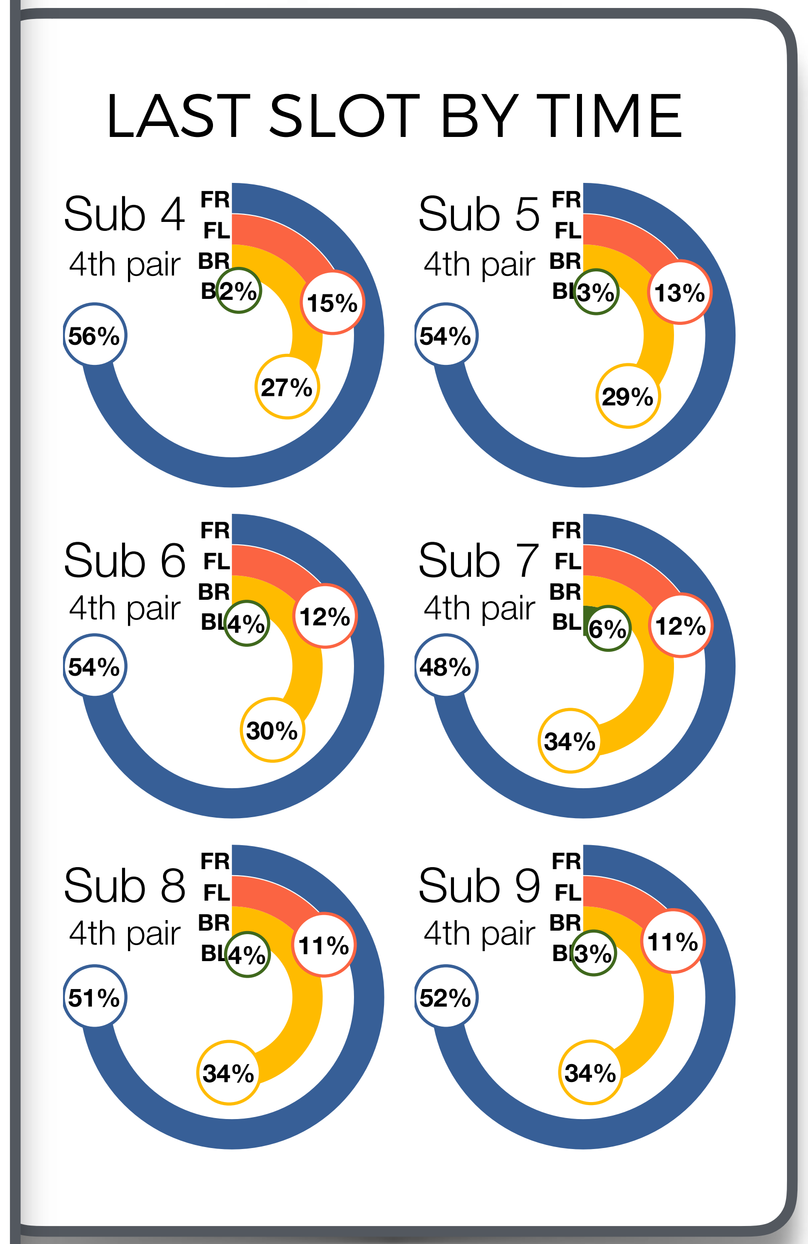
SLOT USAGE BY F2L PAIR



FIRST PAIR SLOT BY SOLVE TIME



LAST SLOT BY TIME



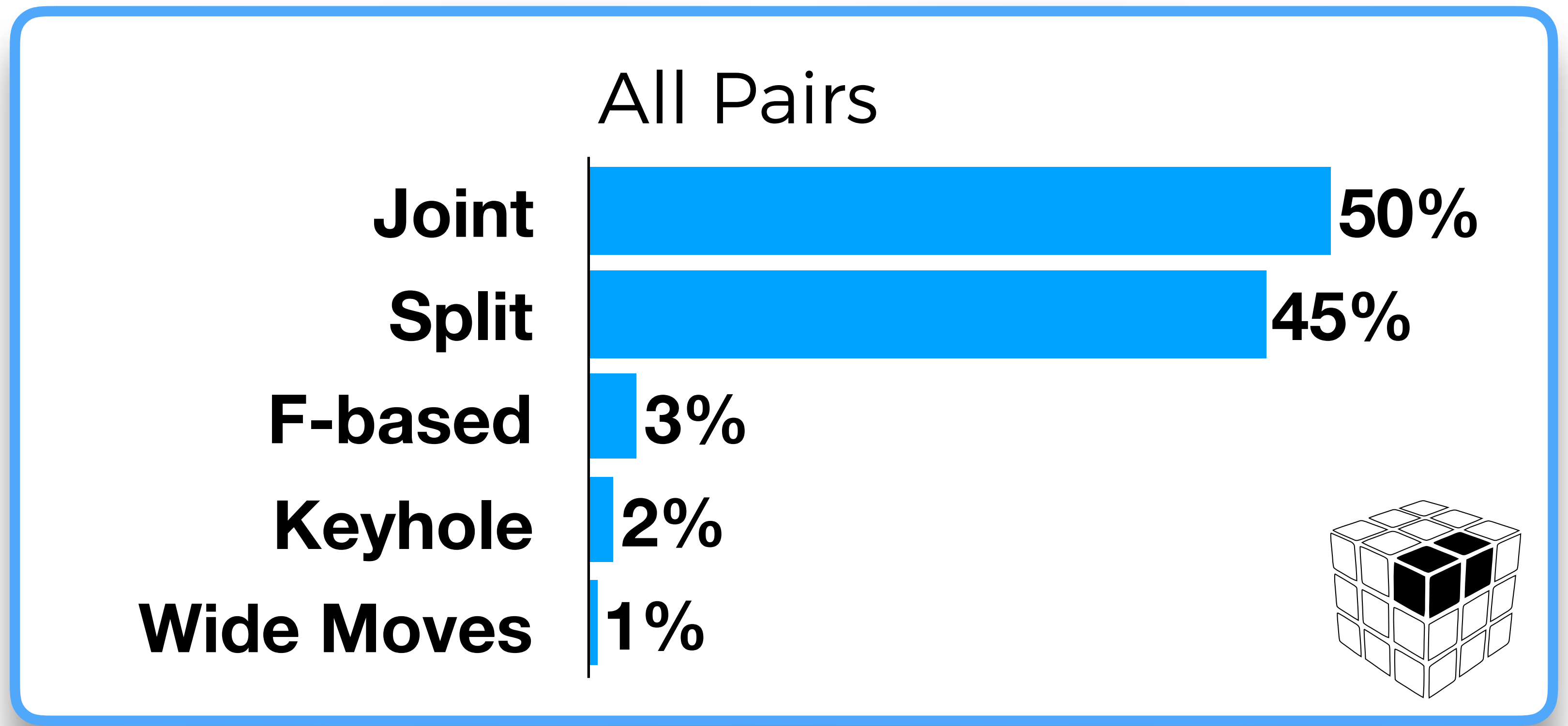
Left Out

Overall for any of the pairs, the left slots comprise less than 40% of slot usage, which means that solvers prefer to fill in on the right and cube rotate rather than go mess with left slots

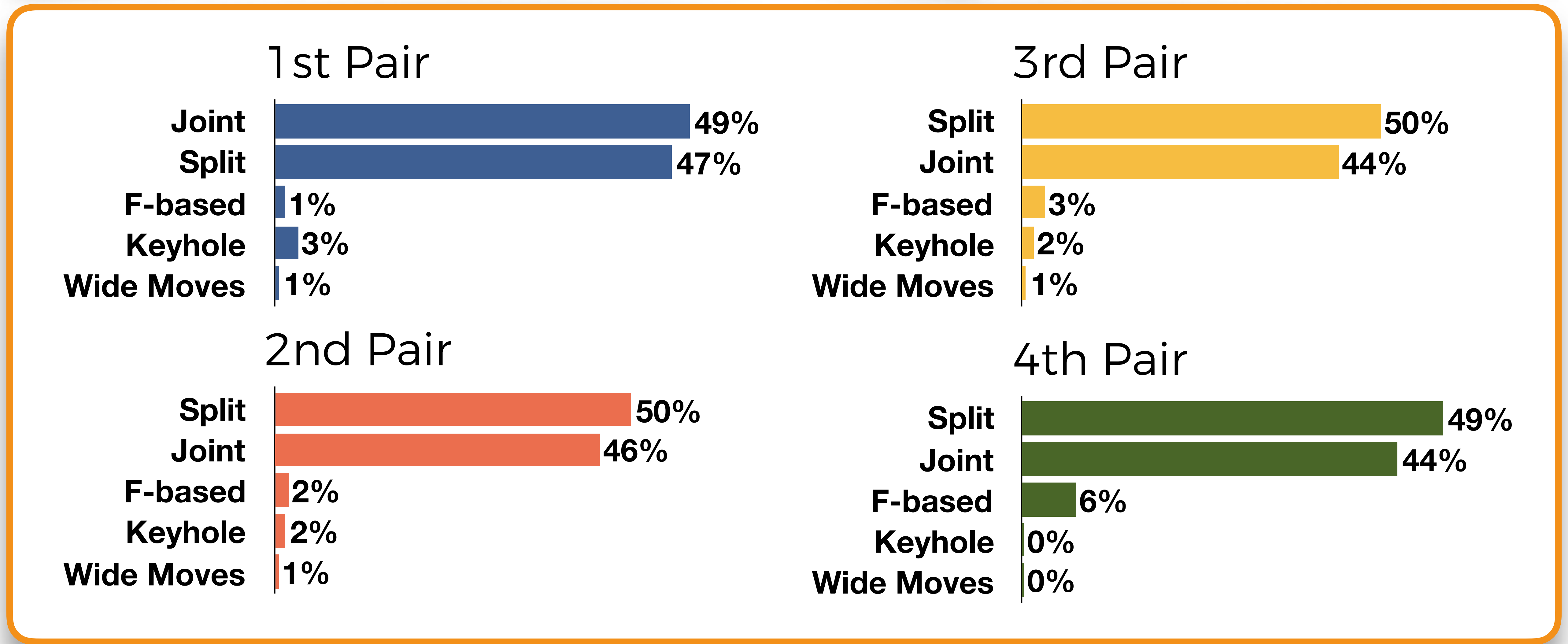
N=4000+

STANDARD INSERTS ARE THE WAY TO GO THE VAST MAJORITY OF TIME, SLEDGE IS USED A BIT MORE ON LAST SLOT, BUT IT REMAINS VERY RARE

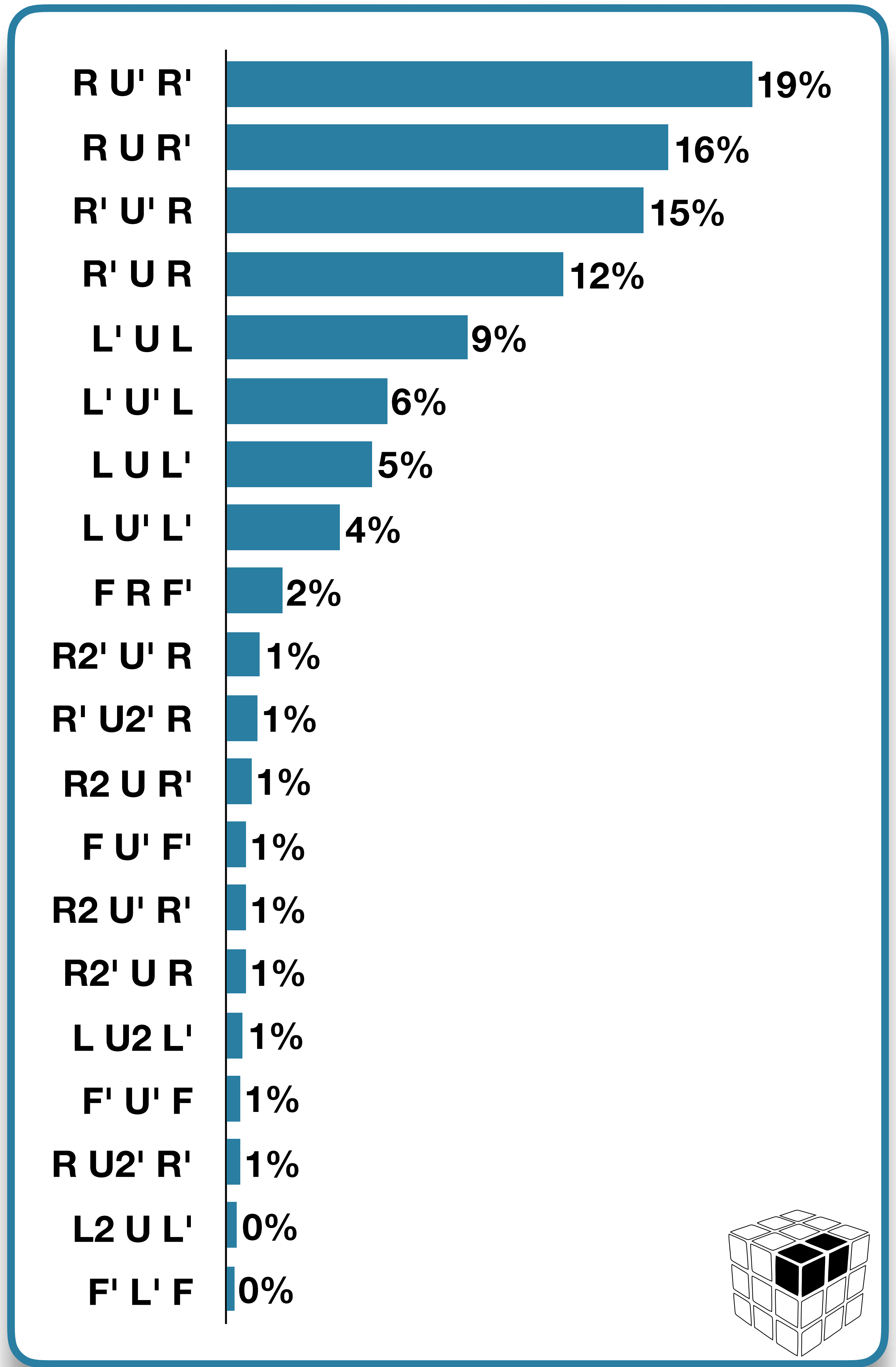
PREVALENCE OF DIFFERENT TYPES OF INSERTIONS



Insertion	for example
Joint	R U R'
Split	R U' R
F-Based	F R F'
Keyhole	D R U' R' D'
Wide	f R f'

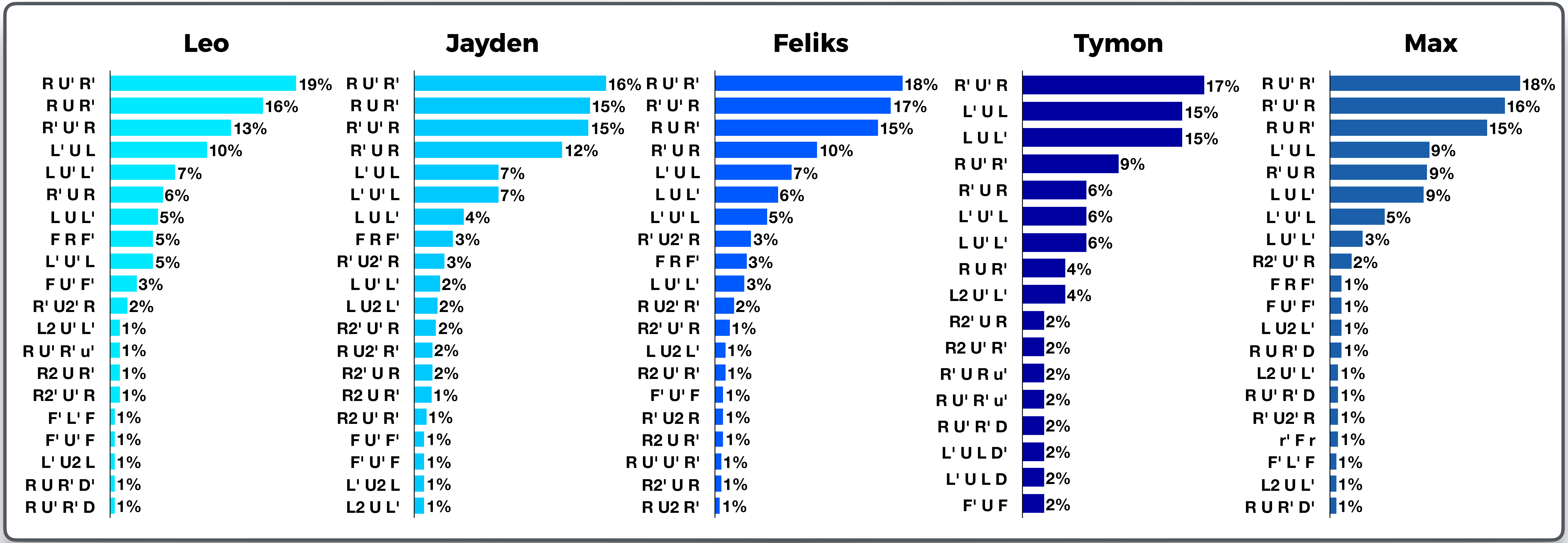


20 MOST FREQUENT INSERTS



DIFFERENT SOLVERS, SLIGHTLY DIFFERENT PREFERENCES

MOST USED INSERTS

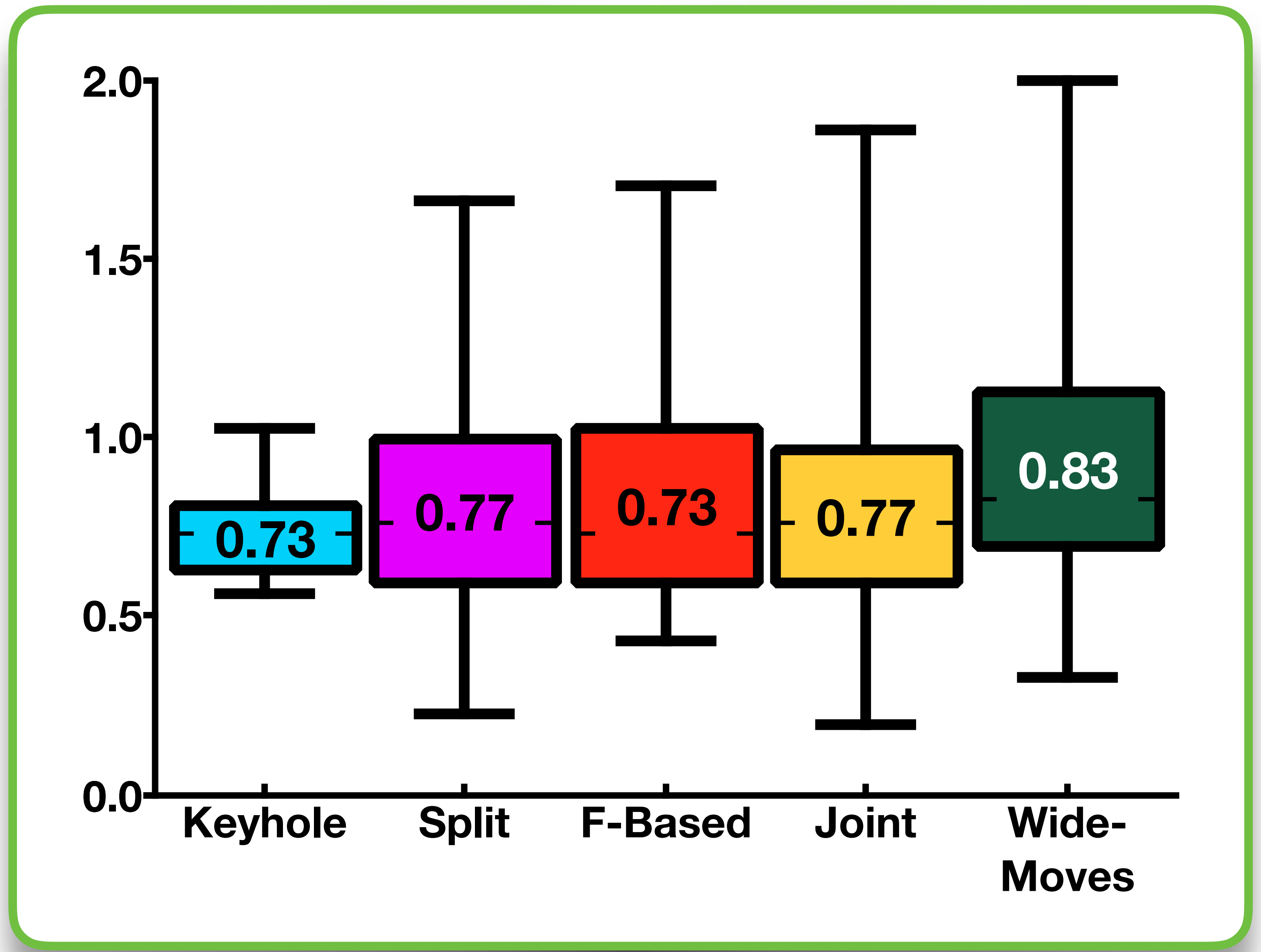


N=4000+

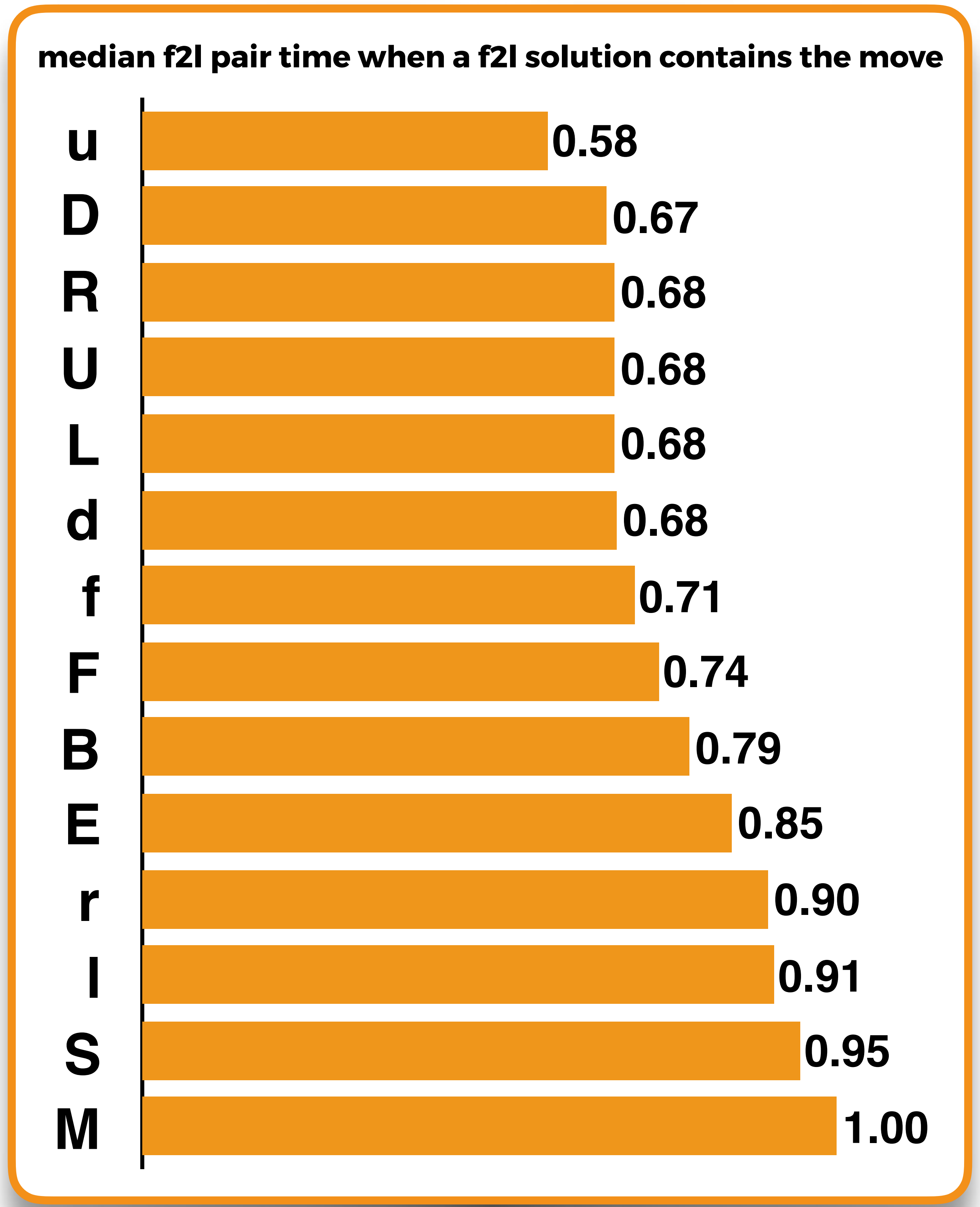
THE INSERT METHOD DOES NOT INFLUENCE THE EXECUTION TIME MUCH, BUT WIDE MOVES AND SLICES DO NOT SEEM TO BE A GOOD IDEA FOR F2L

F2L PAIR TIME BY TYPE

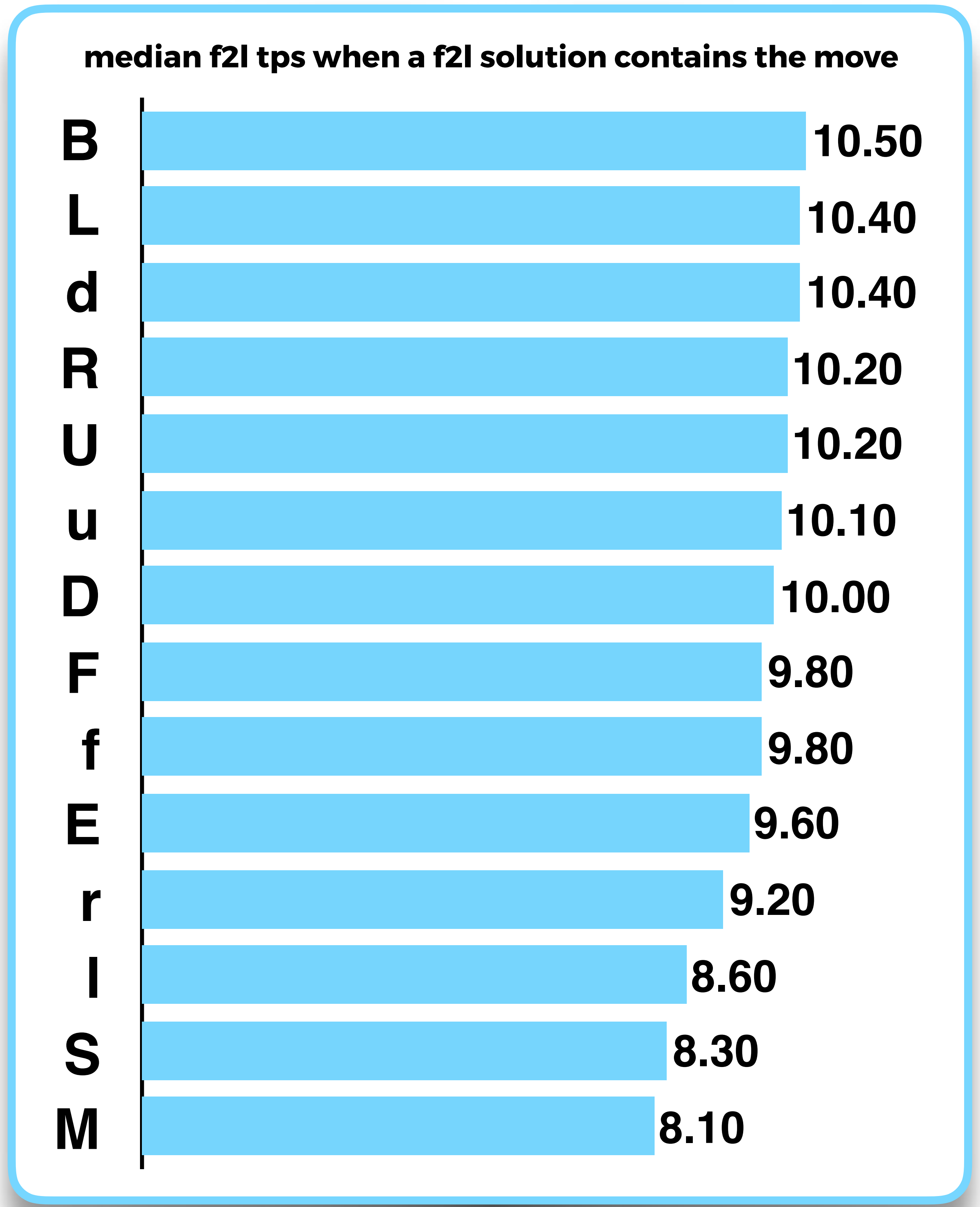
based on 3rd slot, as it is the least affected by cross or LL



F2L PAIR TIME BY MOVE USED

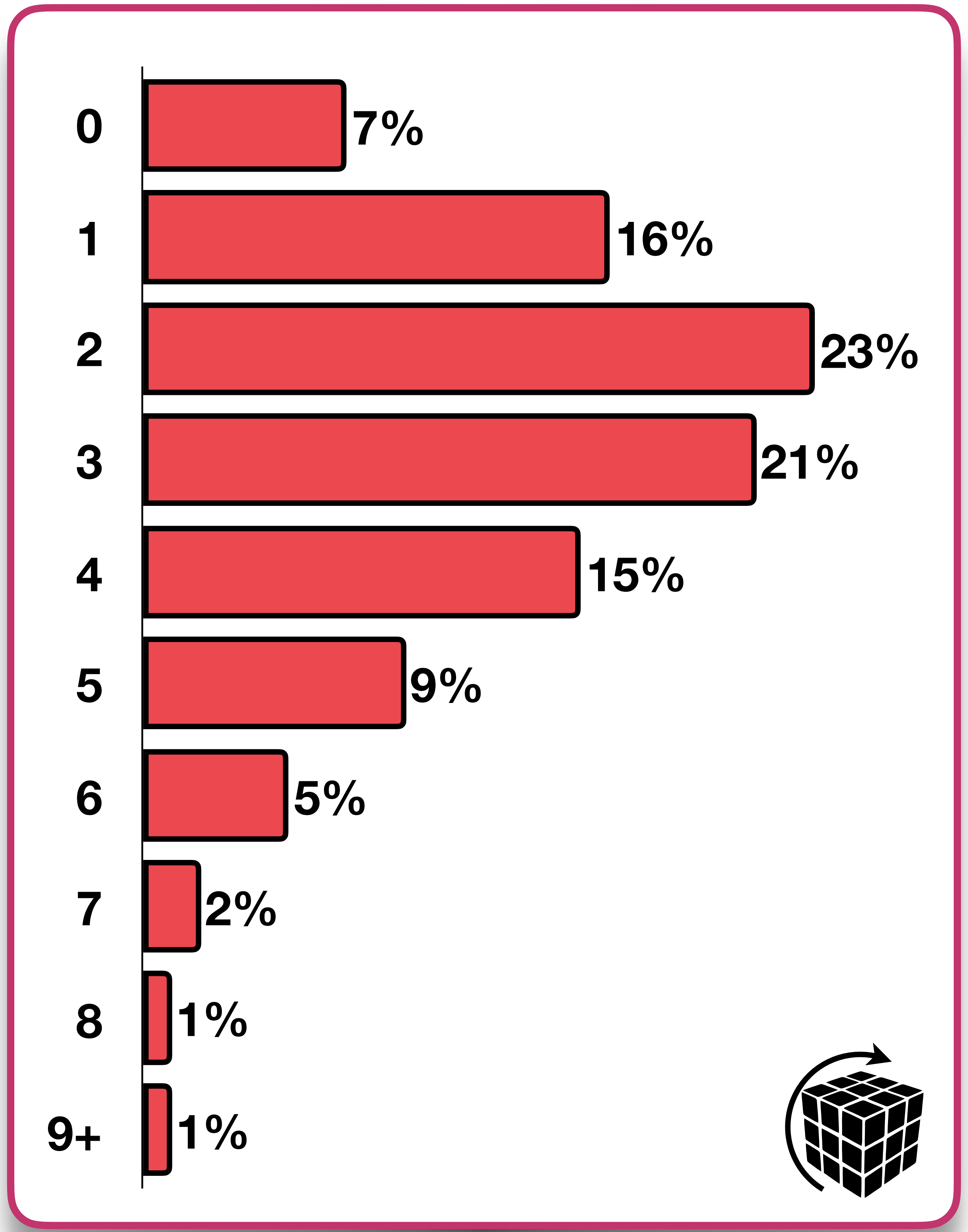


TPS BY MOVE USED

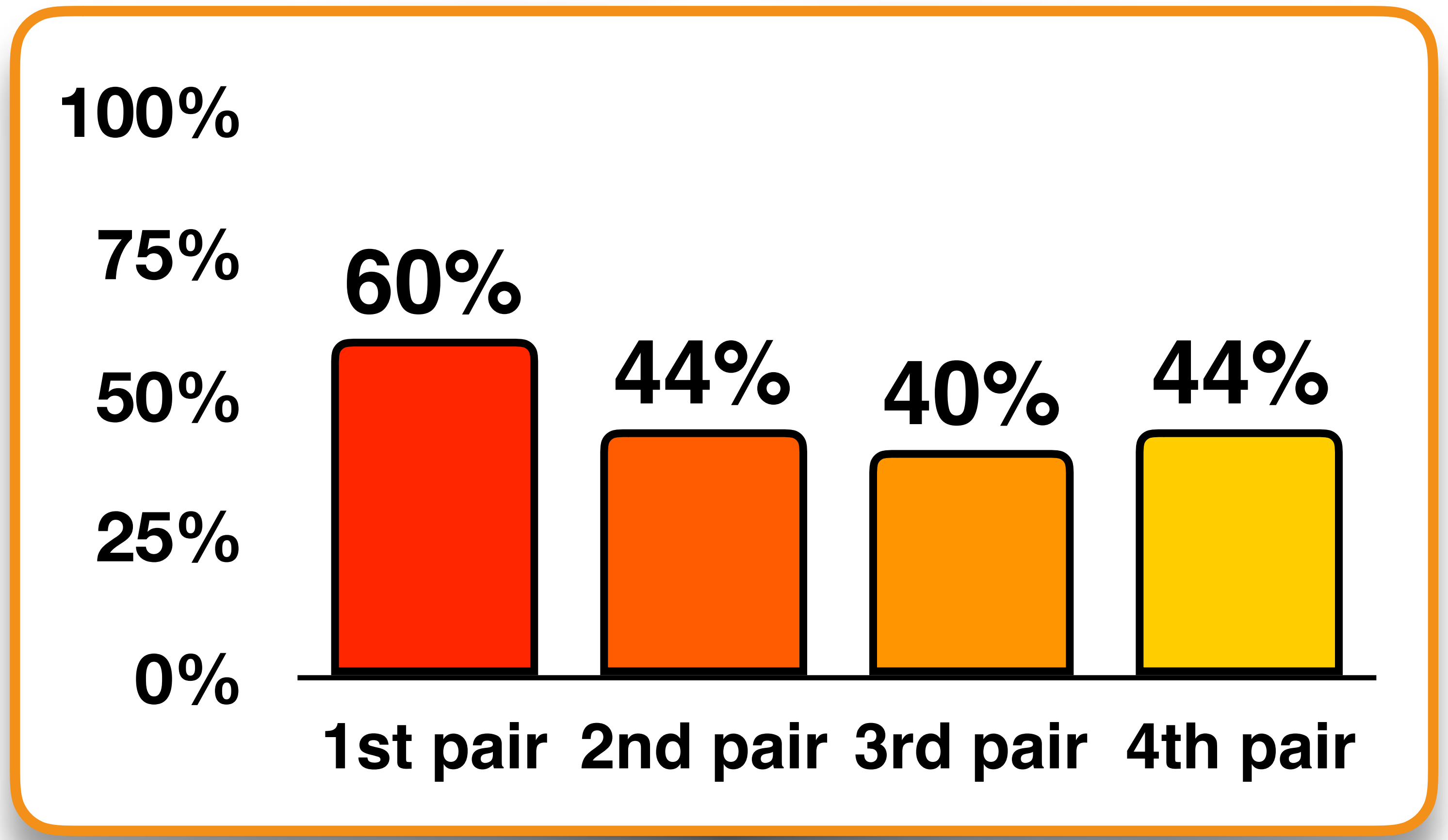


THE TYPICAL F2L HAS 2-3 ROTATIONS, FIRST PAIR IS THE LEAST LIKELY TO NEED ROTATIONS, AND THE FASTER THE SOLVE, THE FEWER THE ROTATIONS

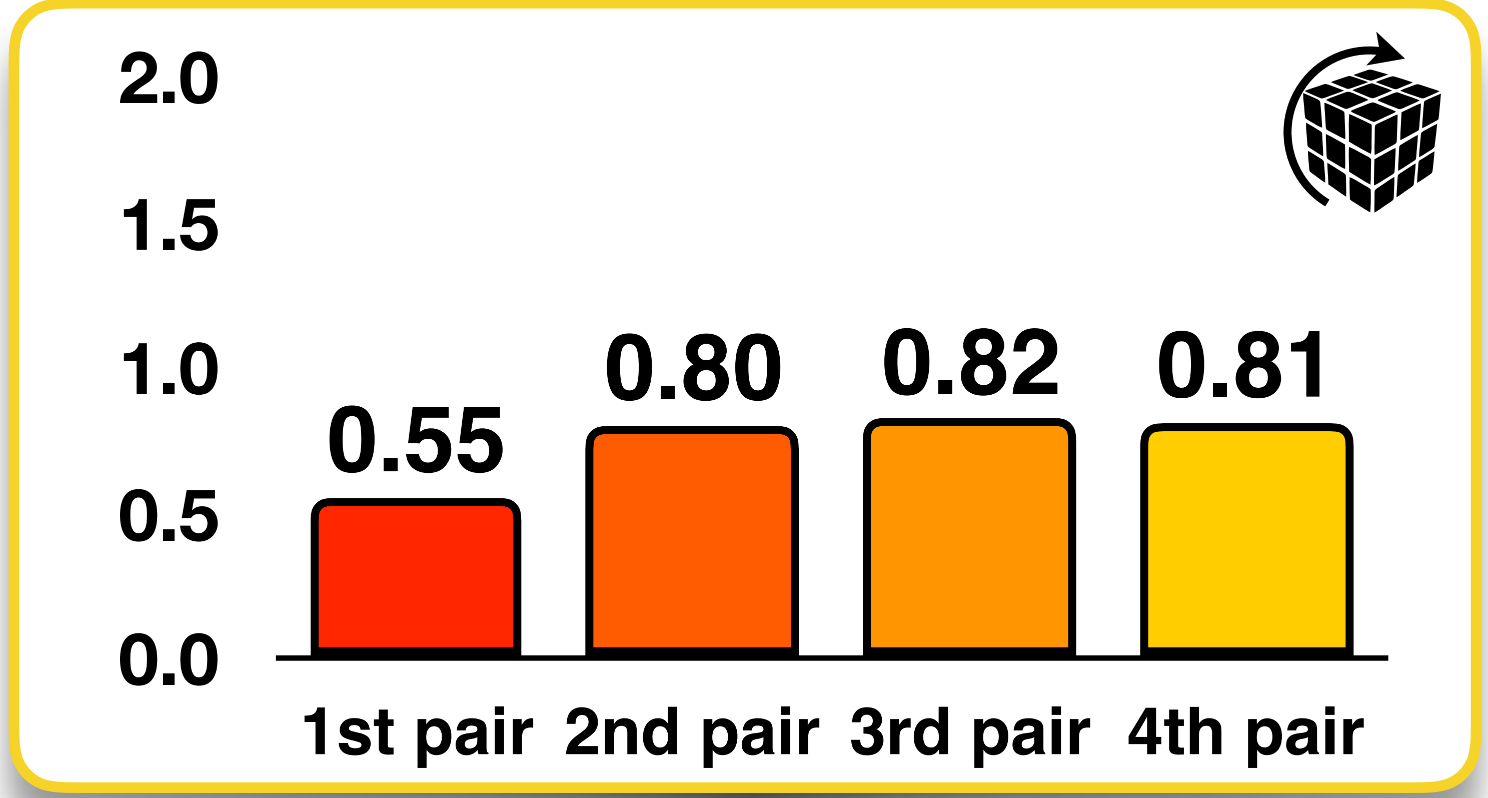
TOTAL ROTATIONS DURING F2L



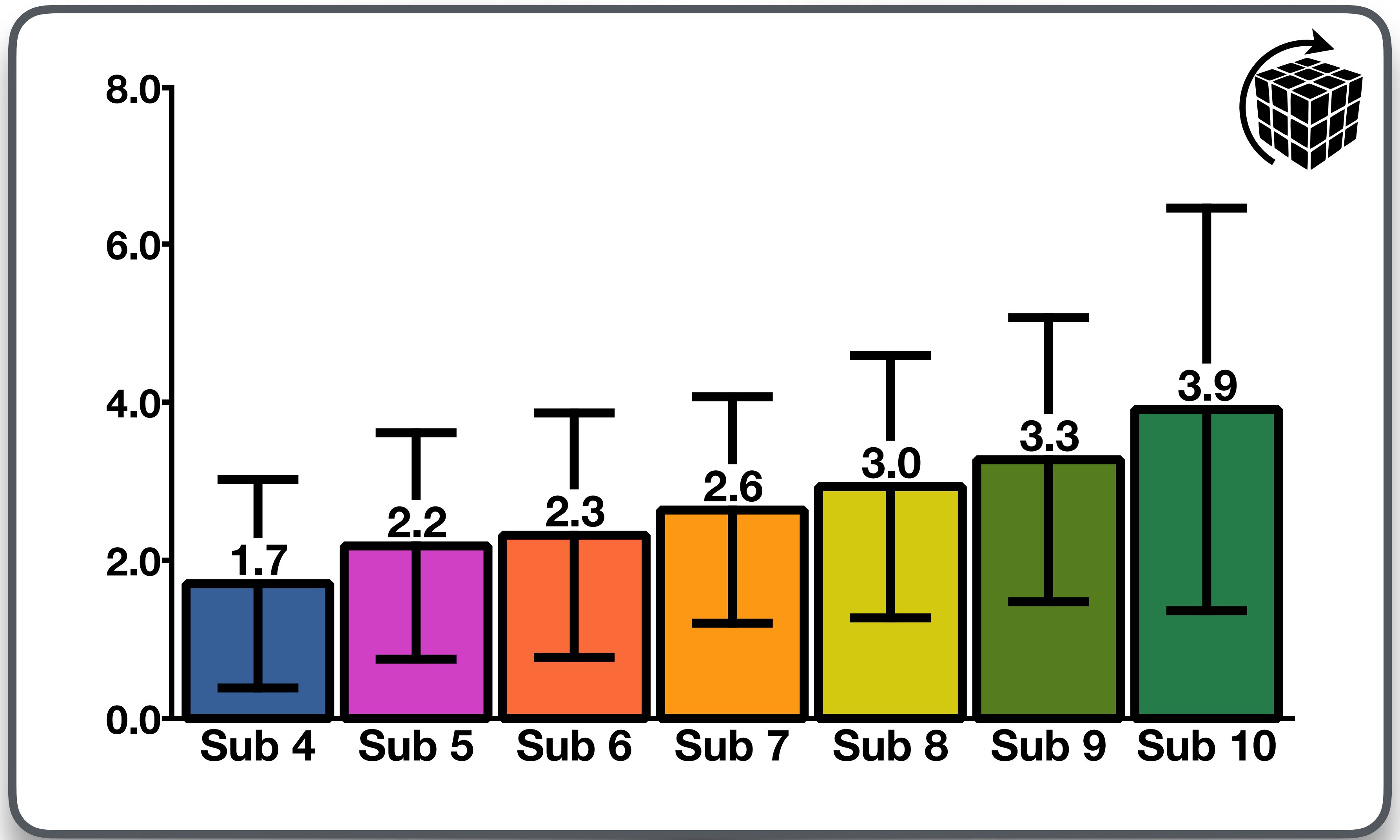
F2L PAIRS WITH NO ROTATIONS



AVERAGE ROTATIONS PER PAIR



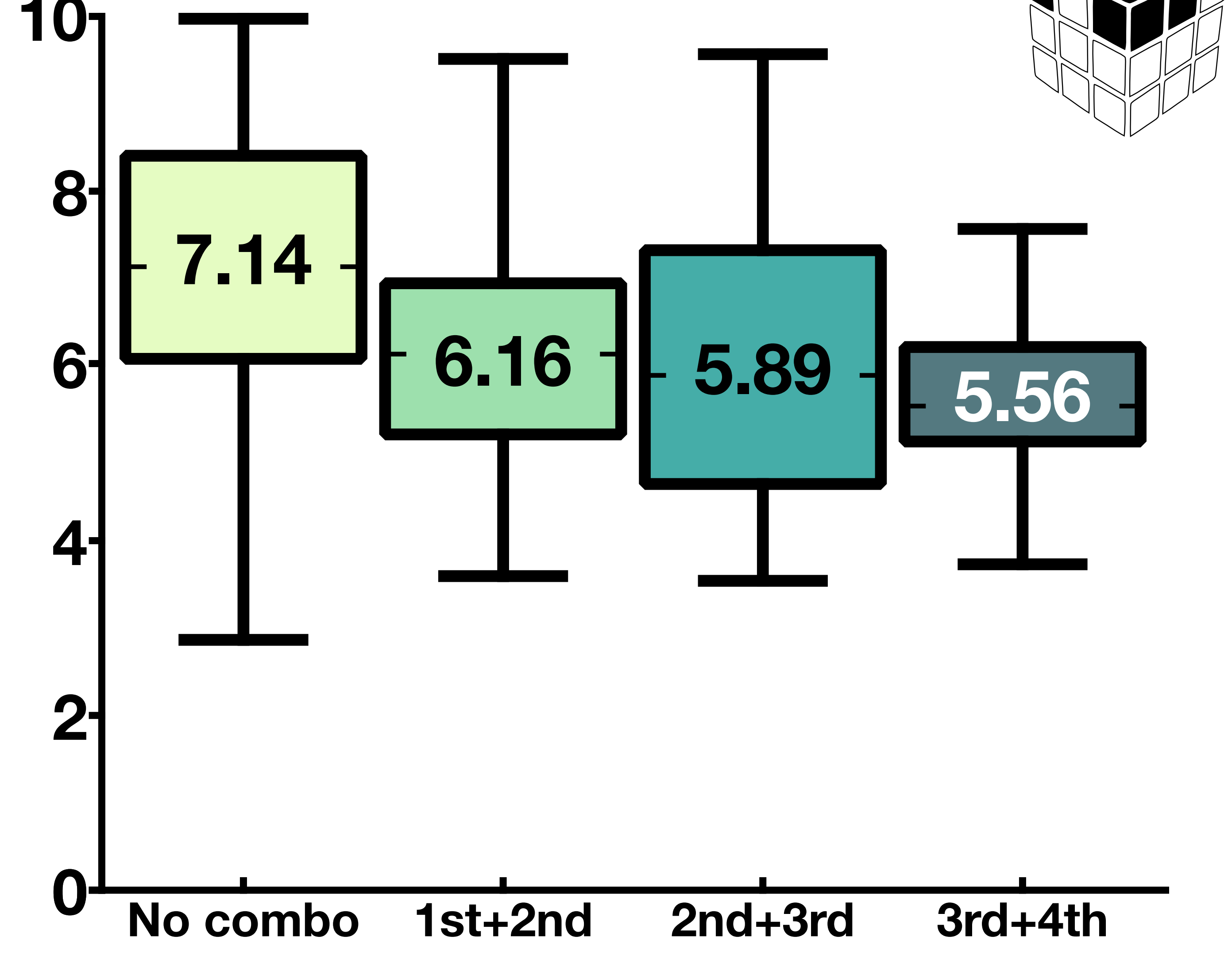
TOTAL F2L ROTATIONS BY SOLVE TIME



N=4000+

COMBO F2L PAIRS : VERY INFREQUENT, BUT THE TIME-SAVES ARE DISCONCERTINGLY HIGH : IS THIS SOMETHING MOST SOLVERS ARE NOT ABLE TO DO?

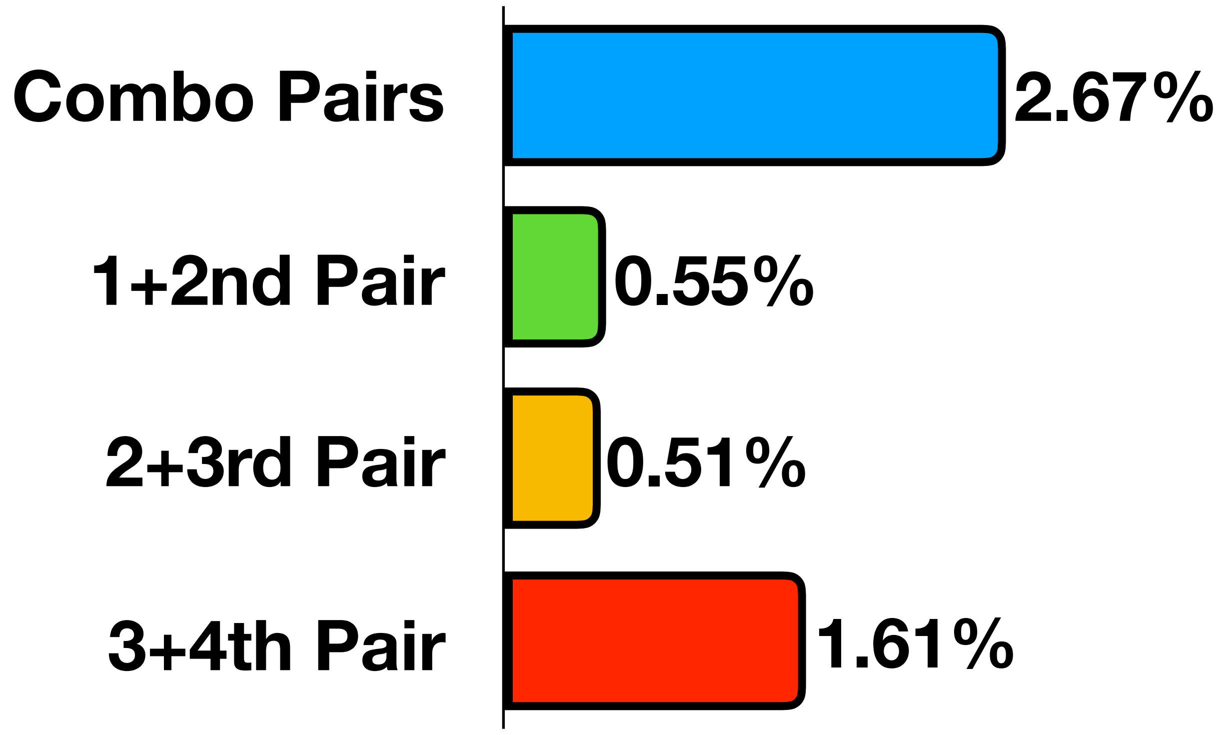
SOLVE TYPE WITH AND WITHOUT COMBO PAIRS



Is there a skill bias?

The difference in solve time cannot be attributed to the time-save of skipping a pair alone: might it be that only the fastest solvers manage to do combos on the fly well?

OCCURRENCES OF F2L PAIRS SOLVED TOGETHER



So rare and yet so powerful

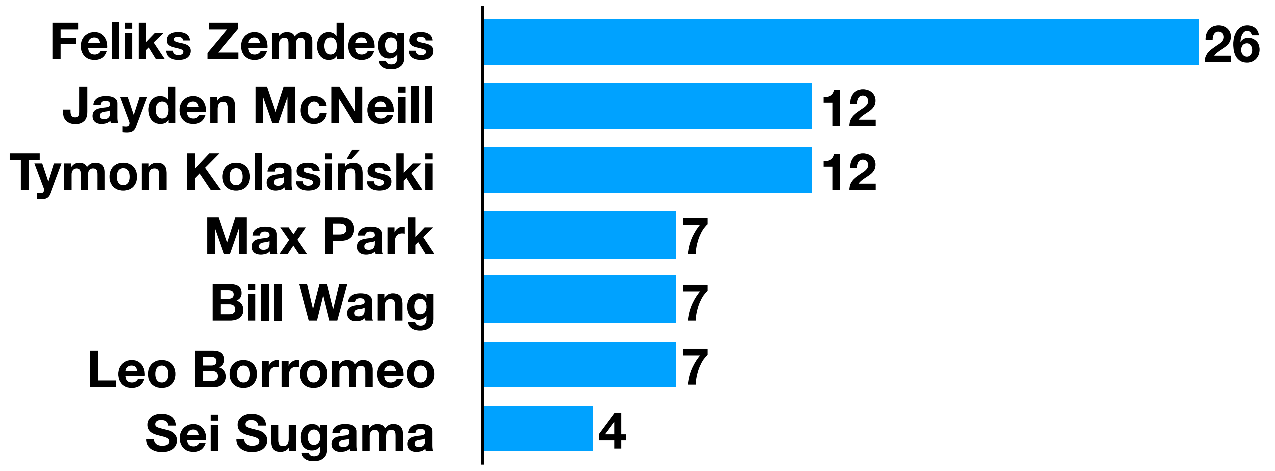
Solving two f2l pairs within the same step is very rare, but when it happens it shaves off significant portions of the total solve

N=4000+

THE KINGS OF PAIR COMBOS : SOME DON'T GAIN A LOT, BUT OTHERS SAVE UP TO 13% OF THE ENTIRE SOLVE

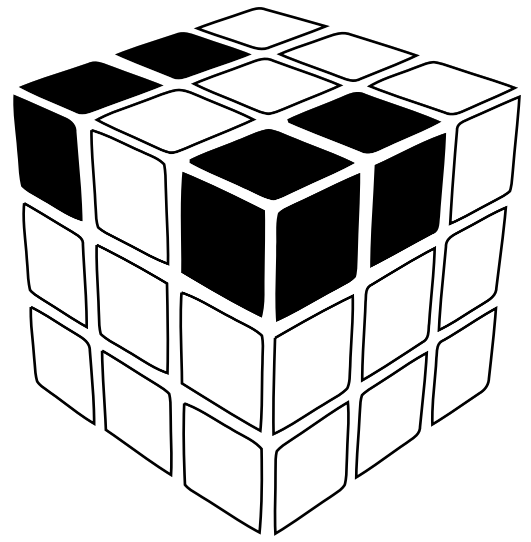
SOLVERS OF F2L PAIR COMBOS

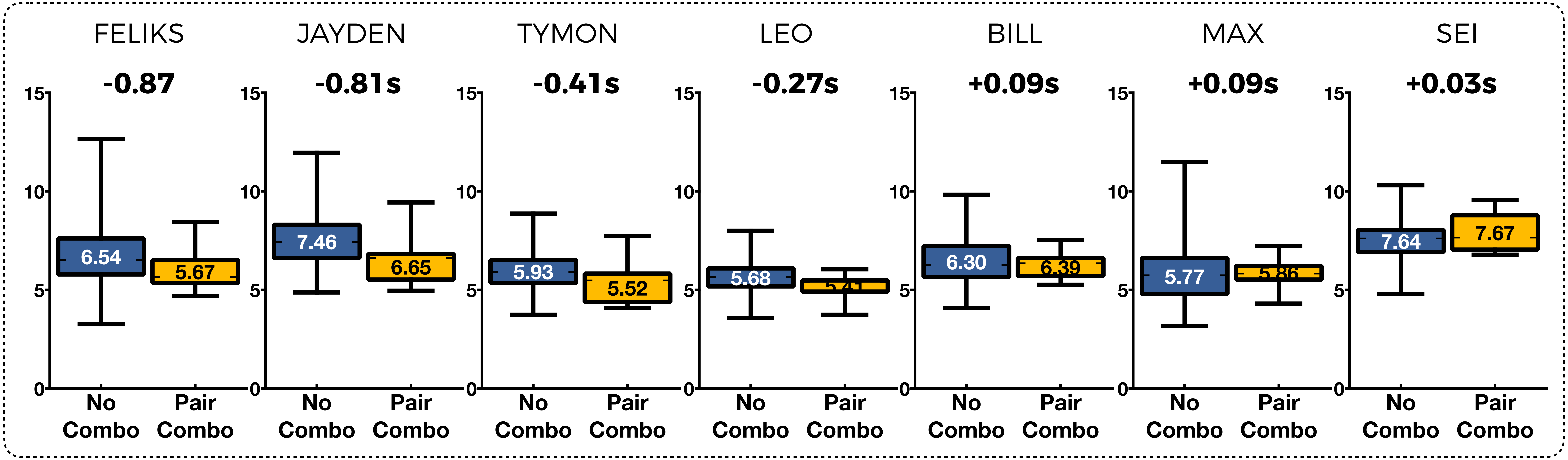
for which we have at 4+ solves with combo pairs and 50+ solves total



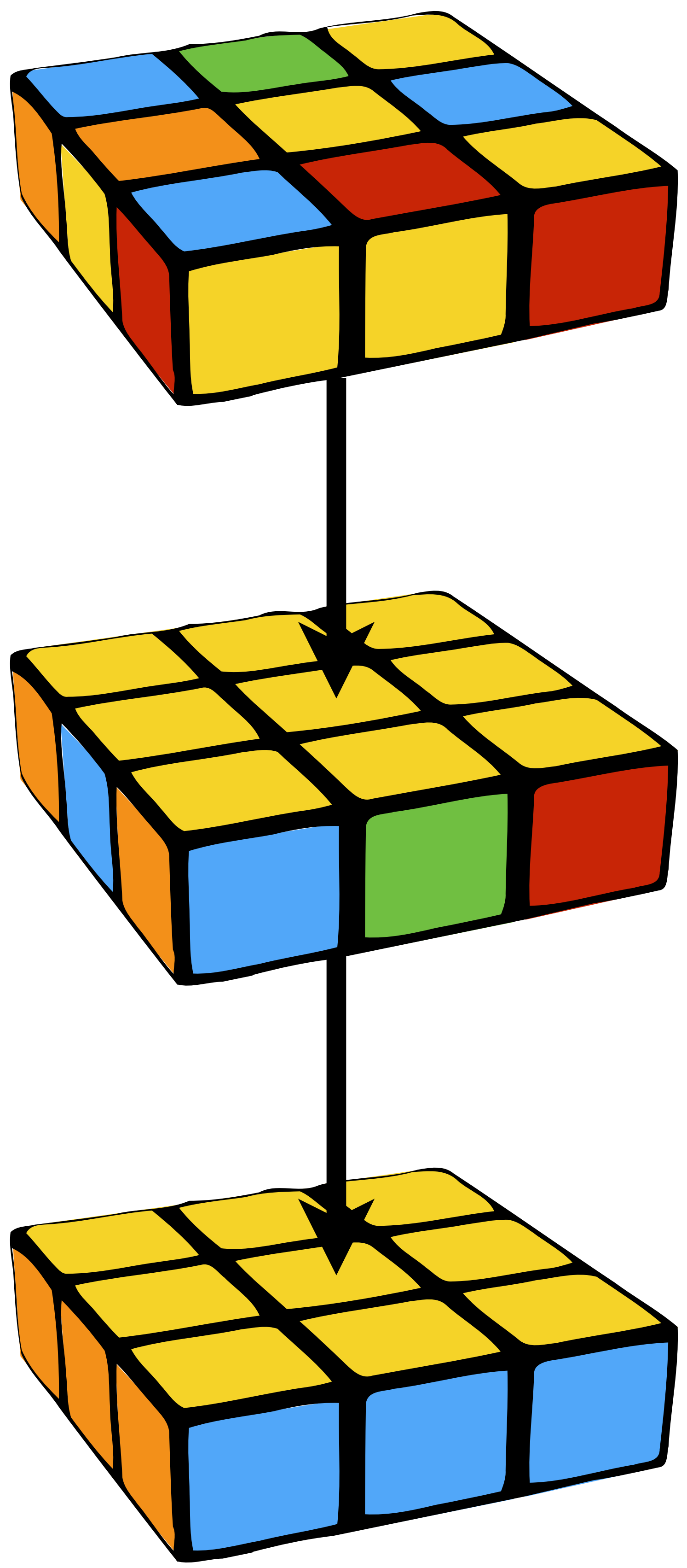
AVERAGE GAIN
(weighed average)

-0.50sec





N=4000+



CHAPTER 4 : LAST LAYER

LAST LAYER IS WHERE A LOT OF THE WORK OF SOLVE OPTIMISATION (OR LUCK) COMES INTO PLAY

OLL

Exec. Time:
1.13 sec

Moves:
10.5 moves
incl. skips

Gens:
3.1 gen

Rotations:
0.25 rot.

PLL

Exec. Time:
1.44 sec

Moves:
12.7 moves
incl. skips

Gens:
3.0 gen

Rotations:
0.45 rot.
mostly x or z rotations

SUB4 SOLVES

	OLL	PLL
Time	**0.92** sec	**0.98** sec
Moves	**8.7** moves	**6.3** moves
Gens	**3.0** gen	**2.7** gen
Rotations	**0.02** rot.	**0.10** rot.

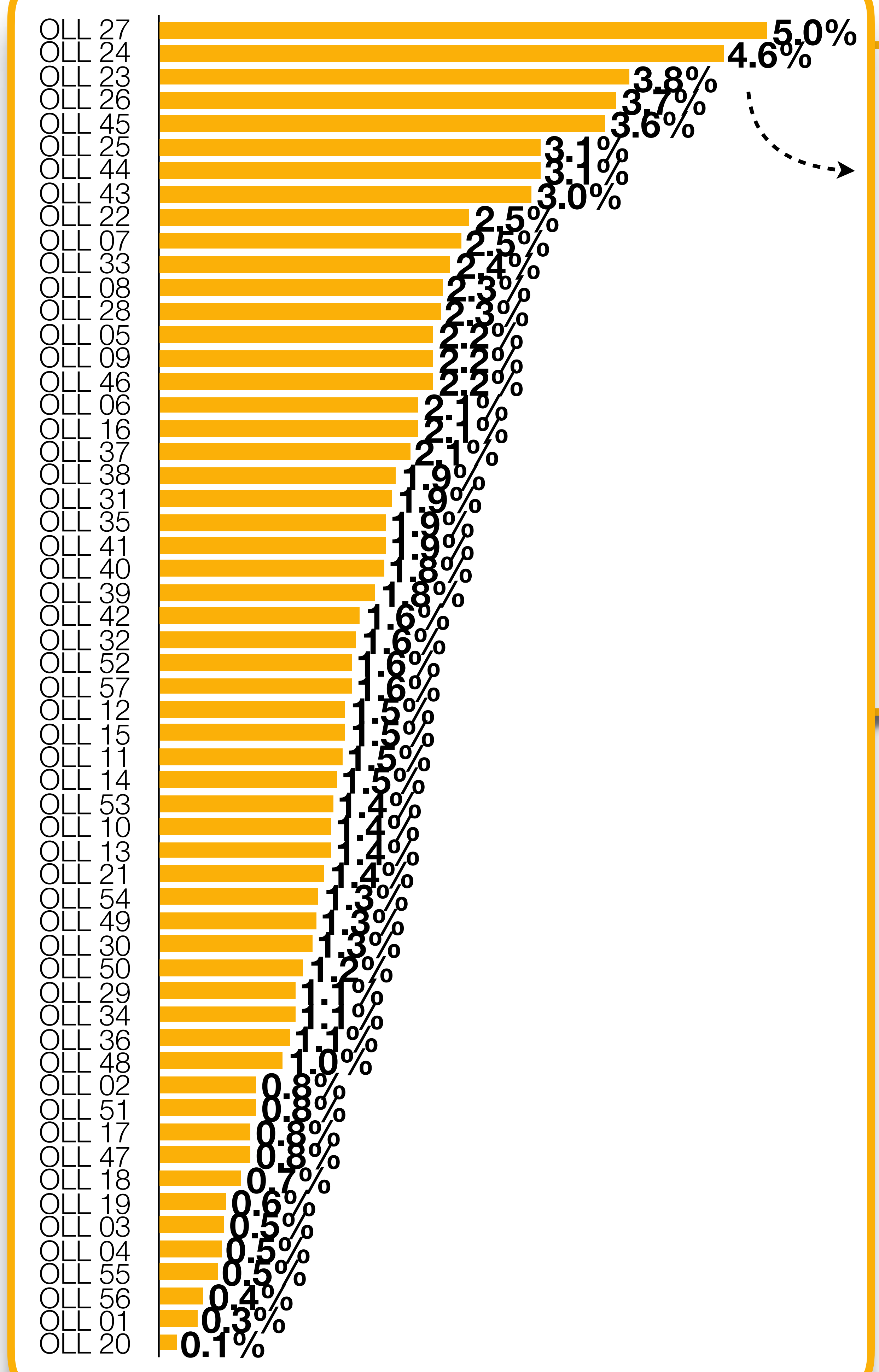
SUB10 SOLVES

	OLL	PLL
Time	**1.23** sec	**1.99** sec
Moves	**10.1** moves	**12.0** moves
Gens	**3.1** gen	**3.0** gen
Rotations	**0.14** rot.	**0.30** rot.

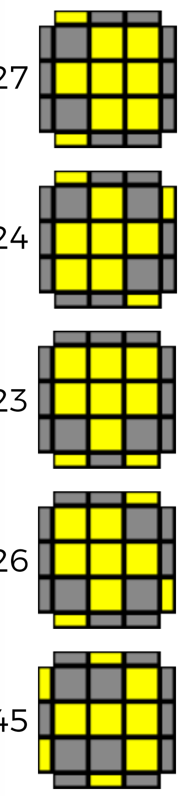
N=4000+

SOME BIG DISPARITIES ACROSS OLLS, WITH THE SLOWEST ONES 2.5X SLOWER THAN THE FASTEST

% FREQUENCY OF OLL CASES

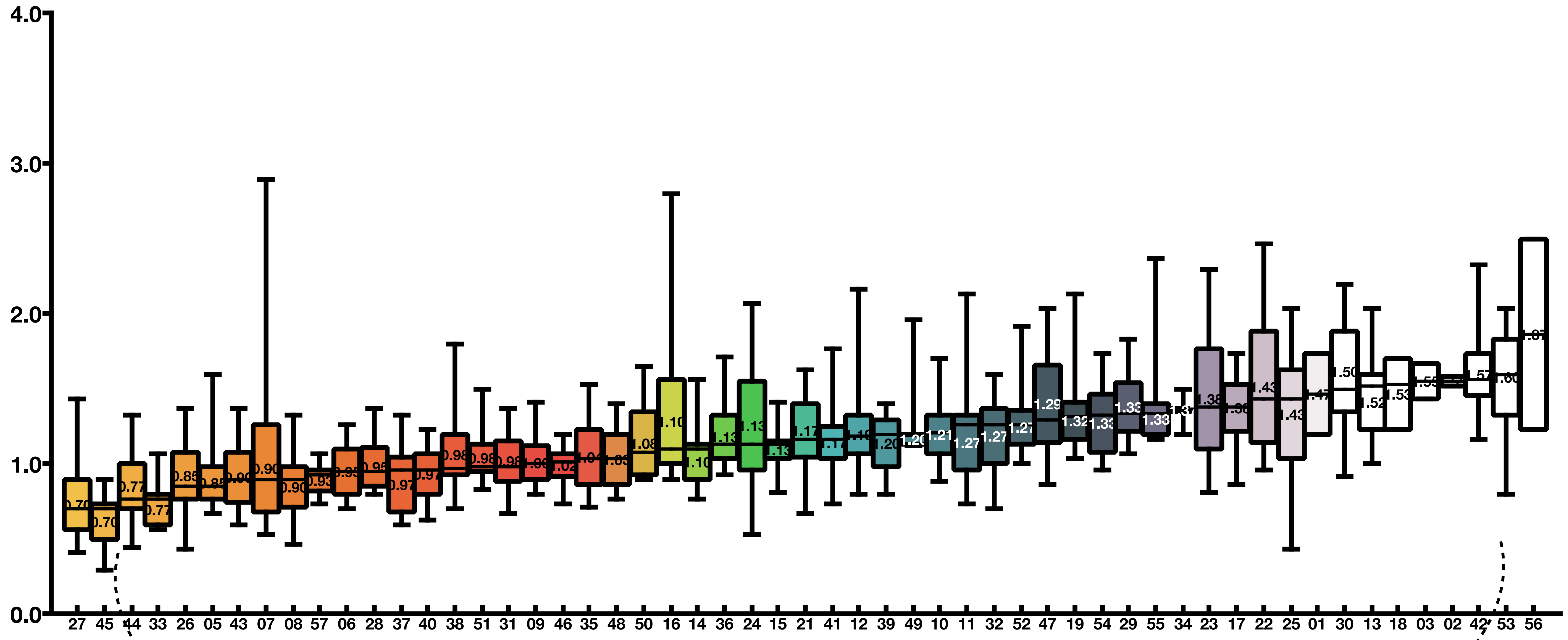


TOP5



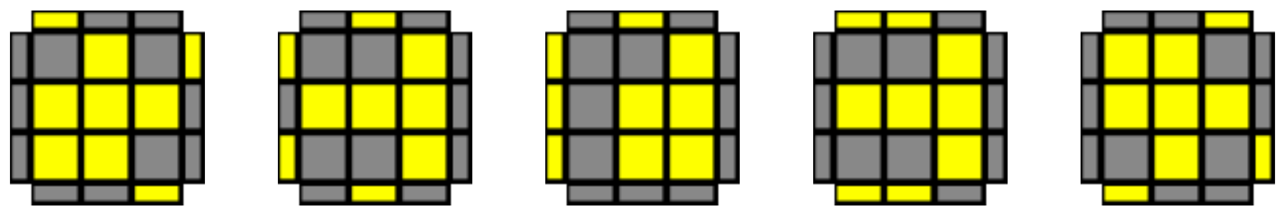
OLL EXECUTION TIME BY OLL CASE

Excludes ZBLL, OLLCP, Skips



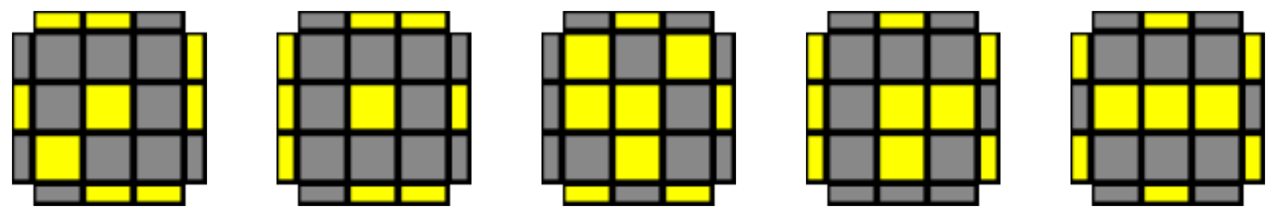
FAST AND FURIOUS 57

5 fastest OLL in live solves



APOCALYPSE NOW

5 slowest OLL in live solves

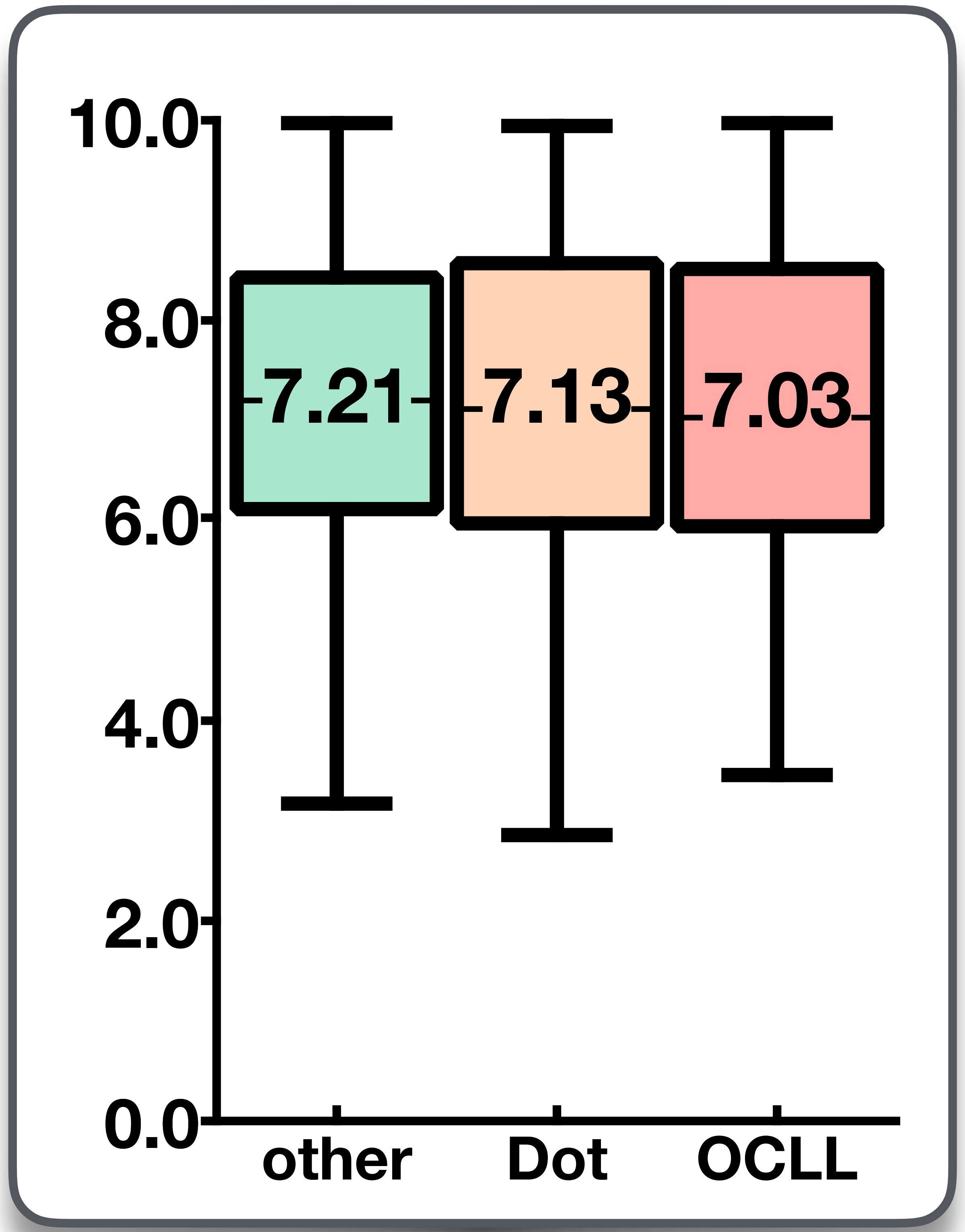


N=4000+

DOT OLLS GET A LOT OF FLAK, AND SOME OF IT IS DESERVED. BUT IS IT WORTH TRYING TO DO SOMETHING TO AVOID THEM? NOT REALLY

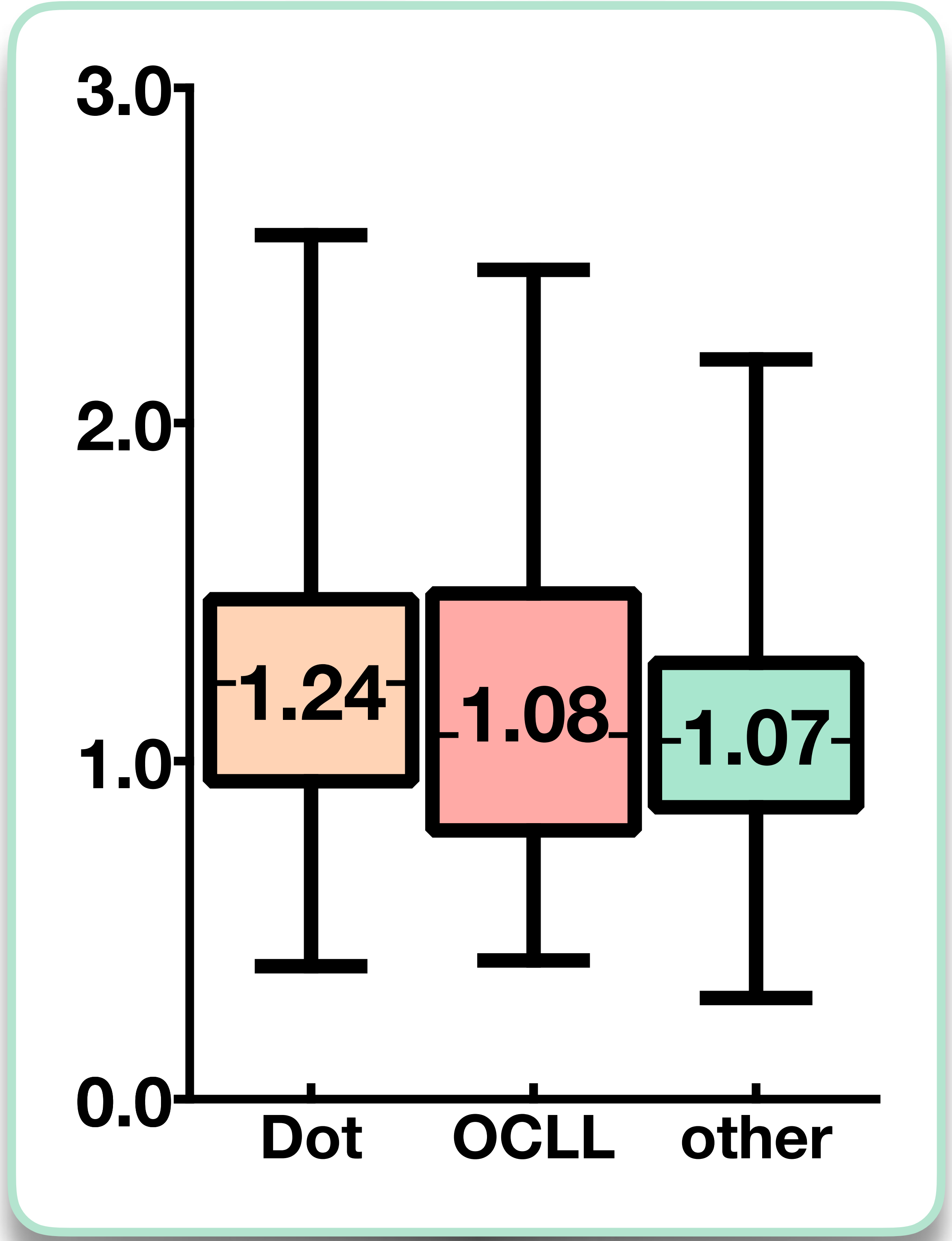
TOTAL SOLVE TIME BY OLL TYPE

Sub10 solves only, incl. skips



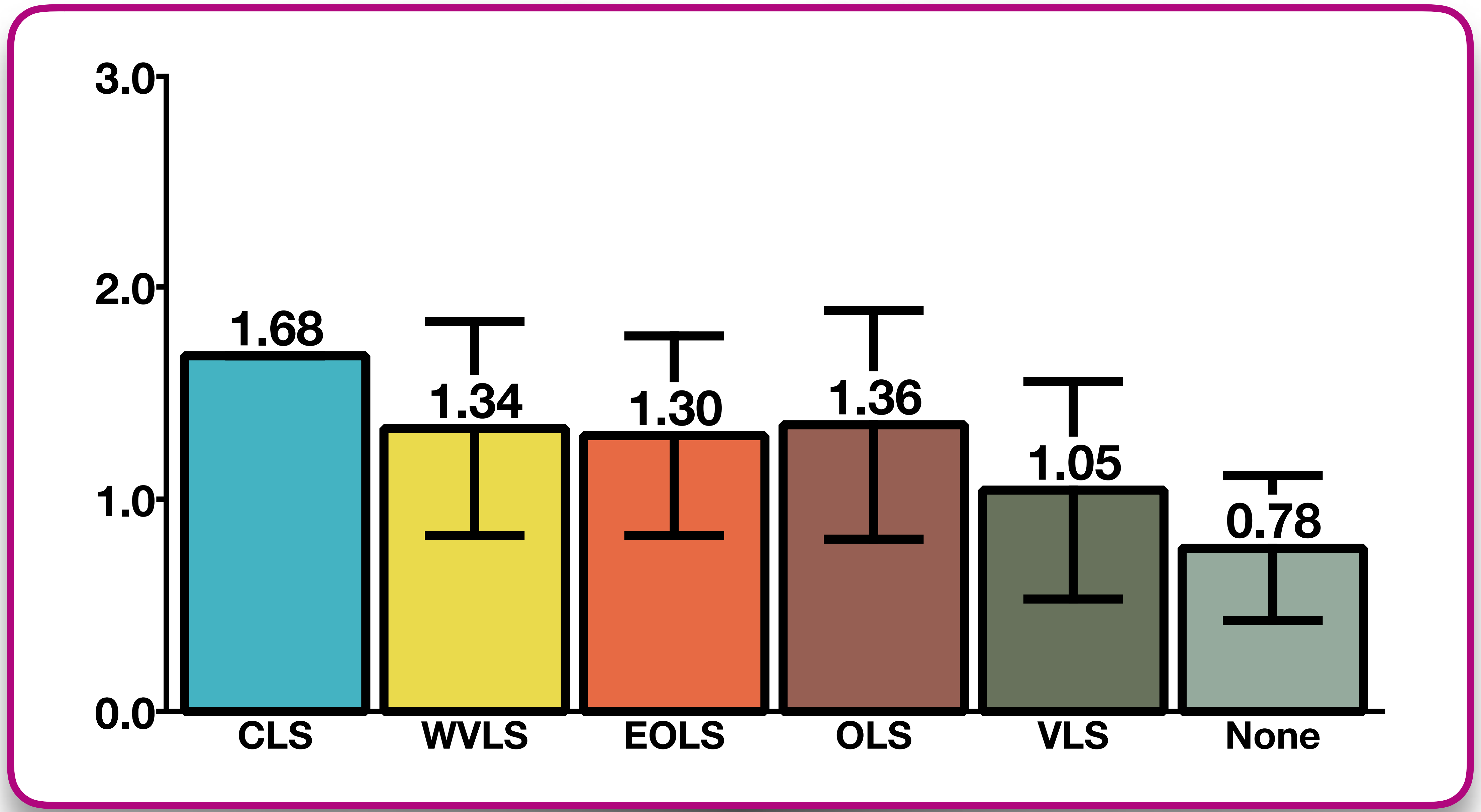
OLL EXECUTION TIME

Excluding ZBLL/OLLCP/skips



LAST PAIR EXECUTION TIME

Sub10 solves only



Median time loss for Dot OLL

0.16 sec

Minimum time loss for Last slot manipulation

0.27 sec

Note: last slot manipulation will still skip OLL a bunch of the time so it's still worth it for that reason!

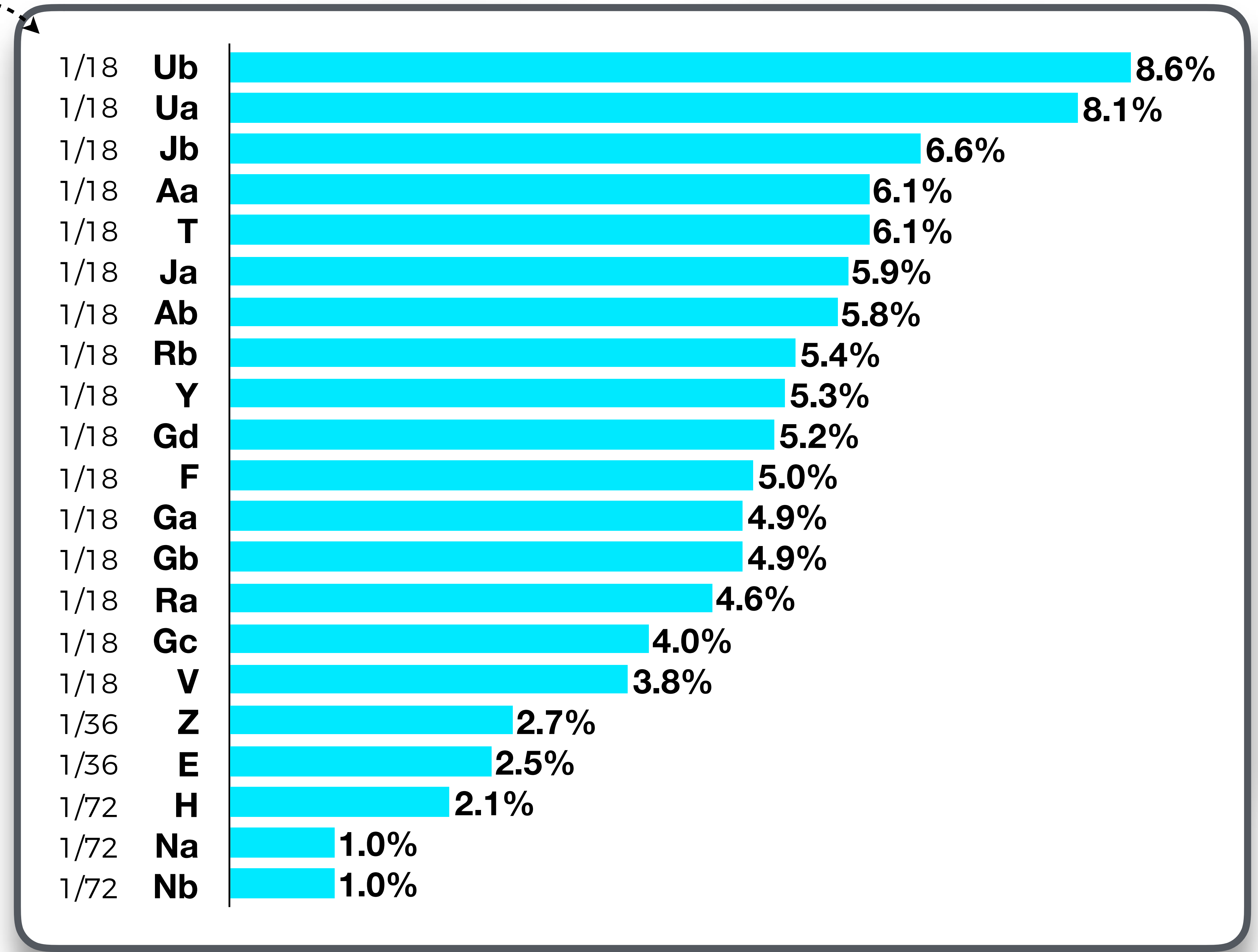
N=4000+

SOME PLLS ARE BORN MORE EQUAL THAN THE OTHERS

% FREQUENCY OF PLL PERM OCCURRENCES

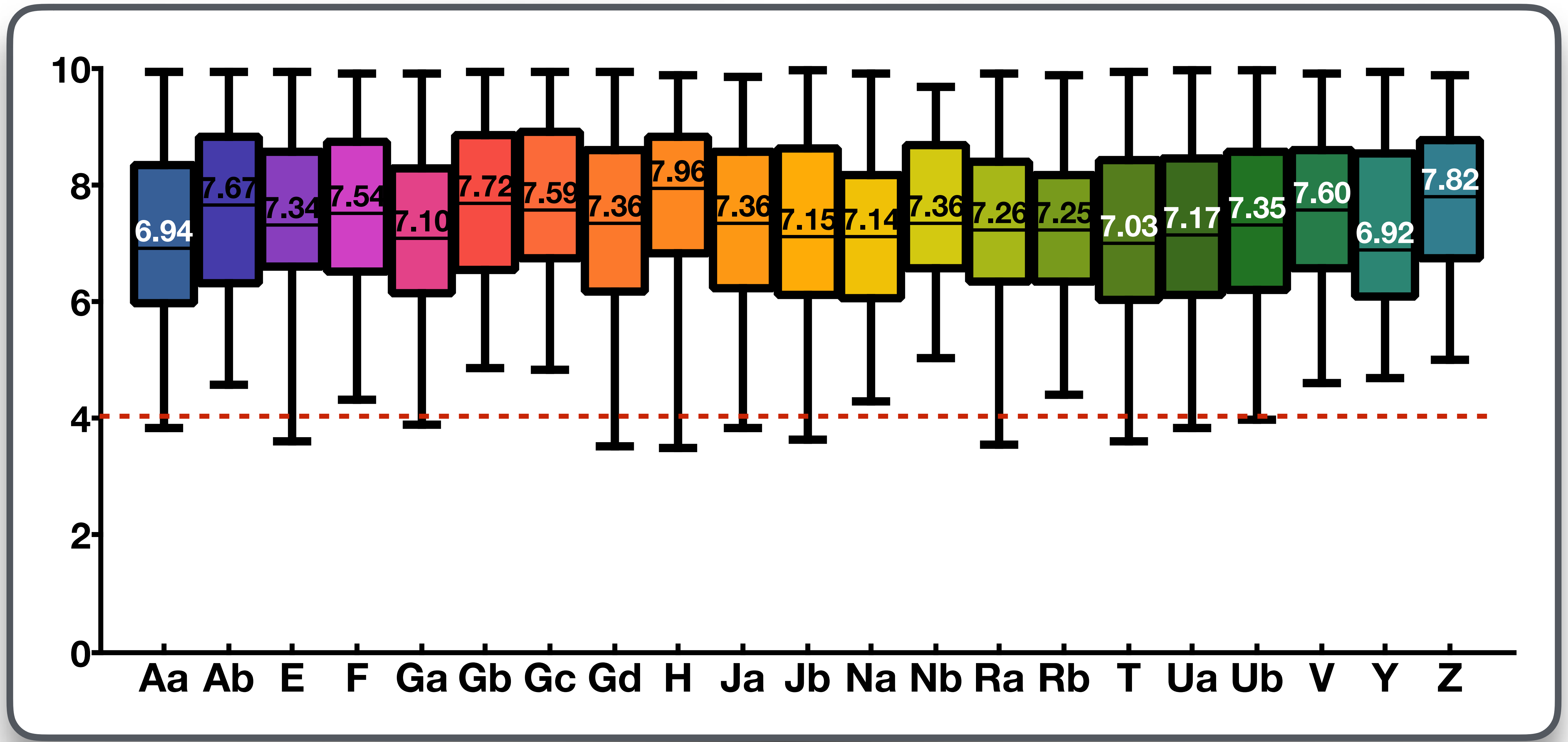
Sub10 solves only, Excluding PLL skips and wonky last layers

NATURAL OCCURRENCES



TIME DISTRIBUTION BY PLL

Sub10 solves only, Excluding PLL skips and wonky last layers



You shall not pass!

Acknowledging that they don't happen often anyway, it is looks like it is not possible for some PLL to be part of a sub4 solve. When looking at the median solve for the different cases, however, the picture is much more mixed, with Y-perm, of all things, coming out on top.

The chicken and the egg

What this data shows is not that solvers are able to influence PLL to the tune of 2x, (although this is also partly the case, especially for EPLL), rather, that it is difficult for a solve to be "good enough" to end up in this database when it had, e.g. a V perm compared to a Jb perm.

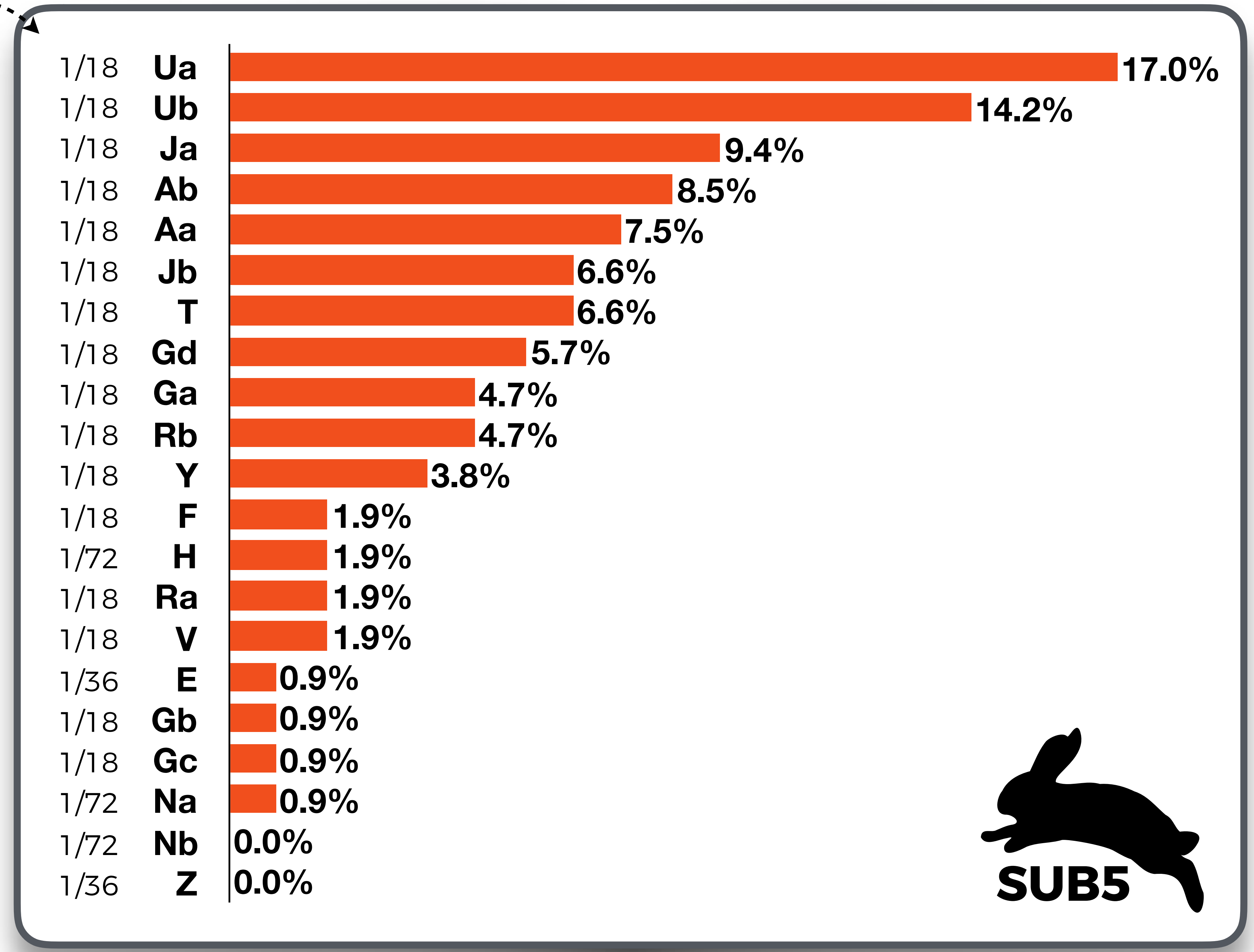
N=4000+

WHAT ABOUT THE FASTEST SOLVES?

NATURAL OCCURRENCES

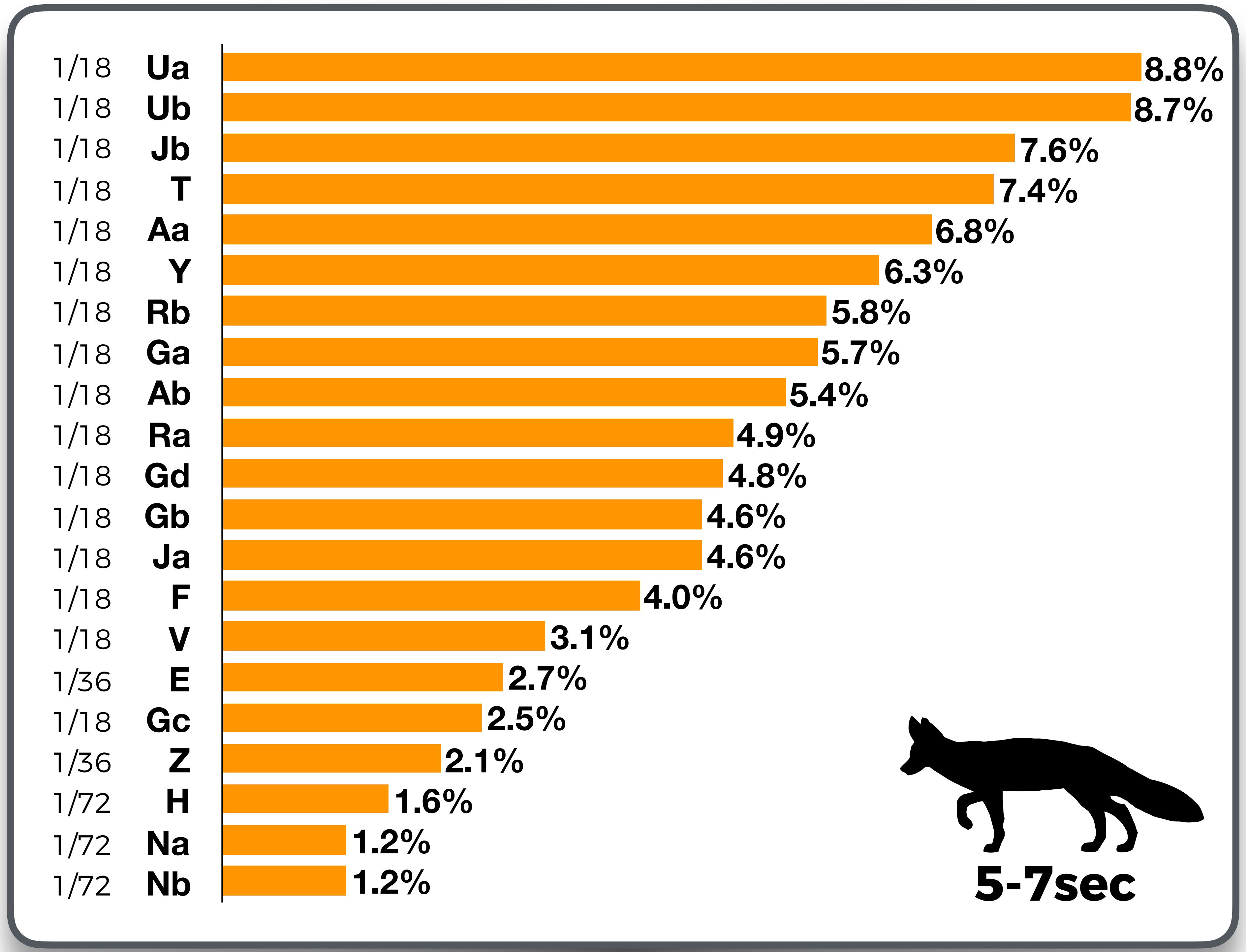
PLL FREQUENCY OCCURRENCES

Sub5 solves only, Excluding PLL skips and wonky last layers



FREQUENCY OF PLL PERM OCCURRENCES

5-7s solves only, Excluding PLL skips and wonky last layers



Note: Sub5 solves without a PLL skip: 36%

Note: 5-7sec solves without a PLL skip: 73%

The case of the lost perms

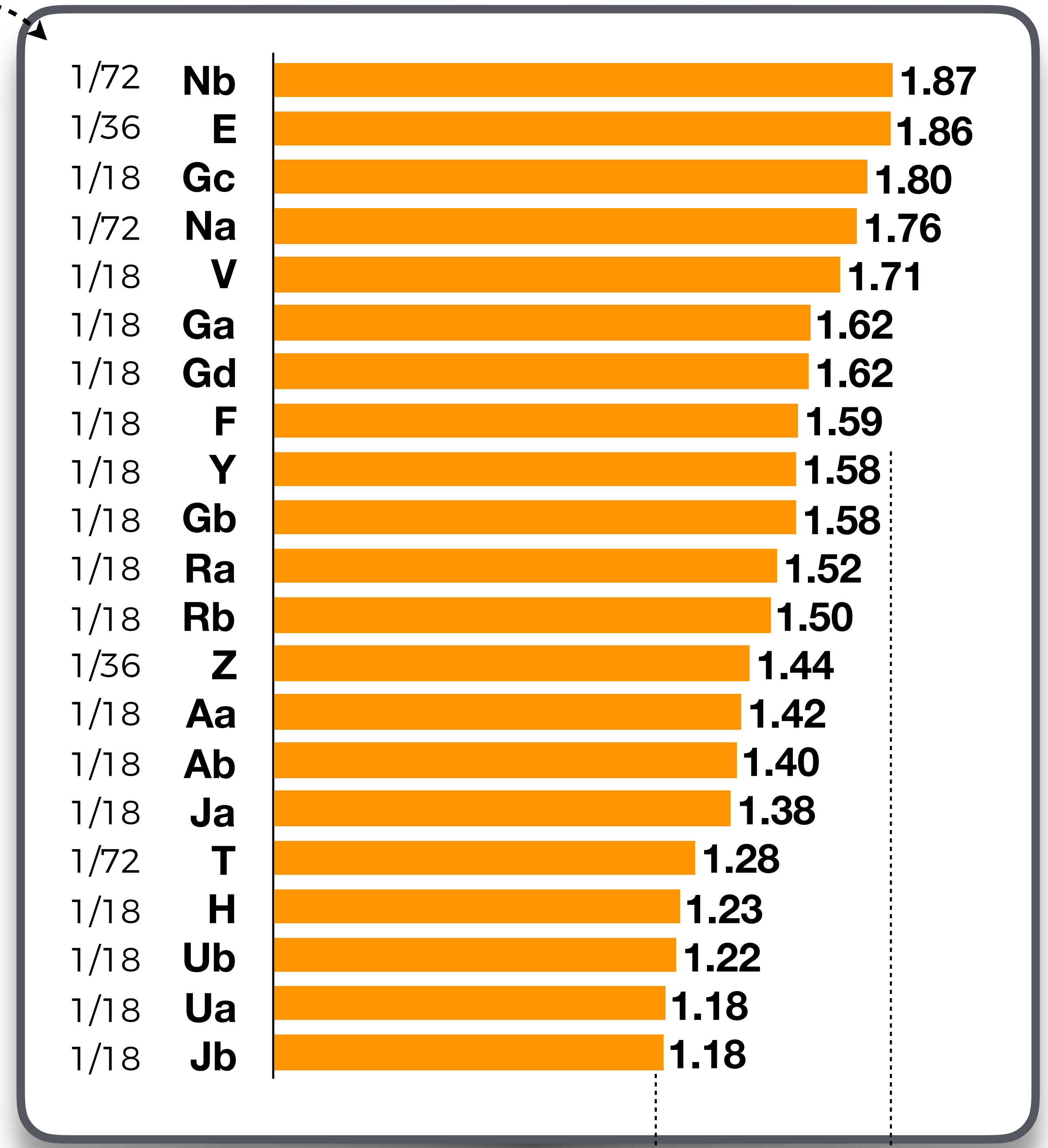
While not necessarily a PB killer, some perms simply disappear from the fastest solves. That said, it is worth remembering that 2/3 of sub5 solves end with a PLL or LL skip

N=4000+

PLL EXECUTION : 1.51 SEC ON AVERAGE, SOMETIMES LESS SOMETIMES MORE

EXECUTION TIME BY PLL TYPE

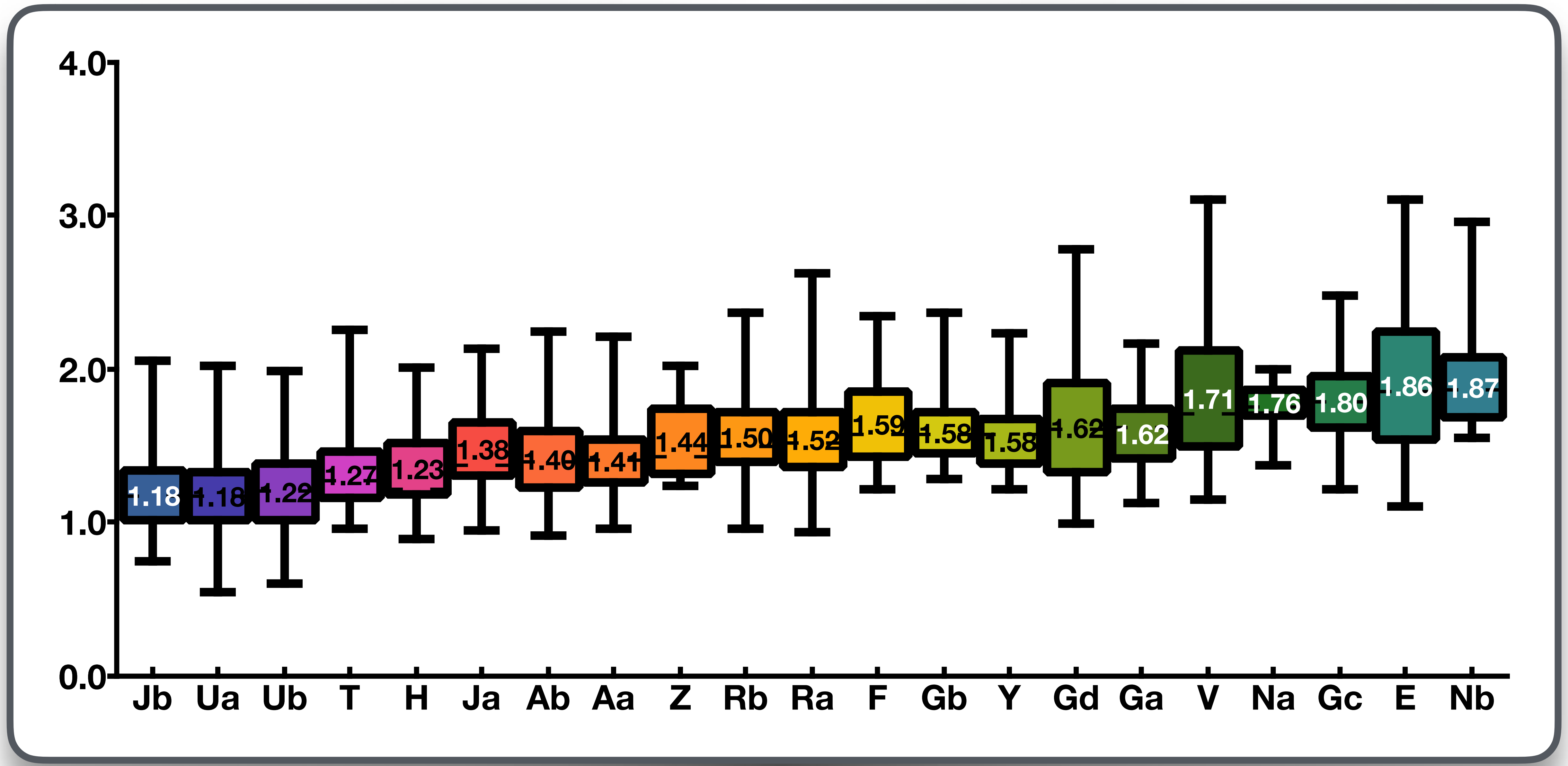
NATURAL OCCURRENCES



Nb vs Jb: +0.69s

DISTRIBUTION OF TIMES PER PLL

Excluding PLL skips and wonky last layers

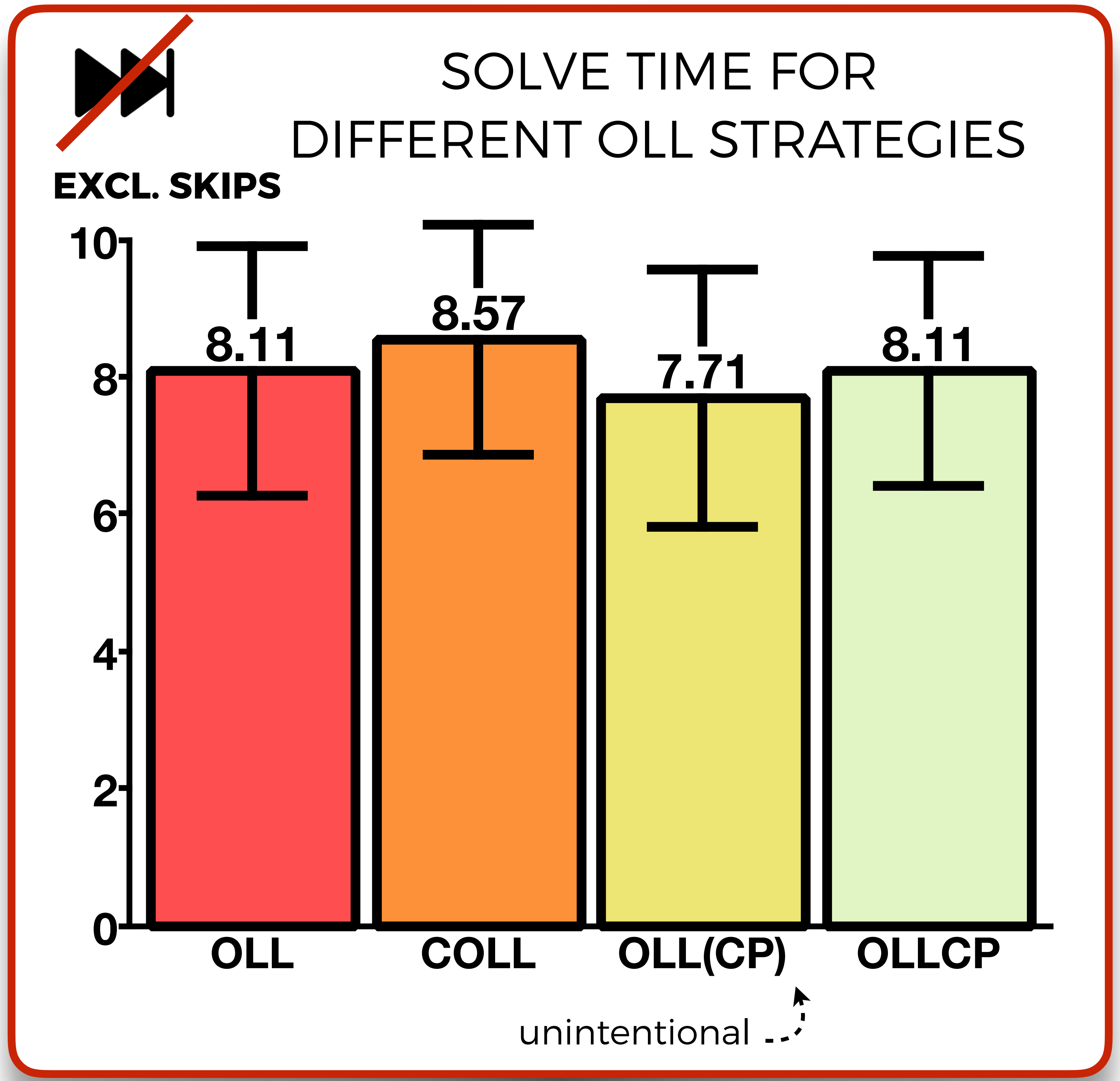


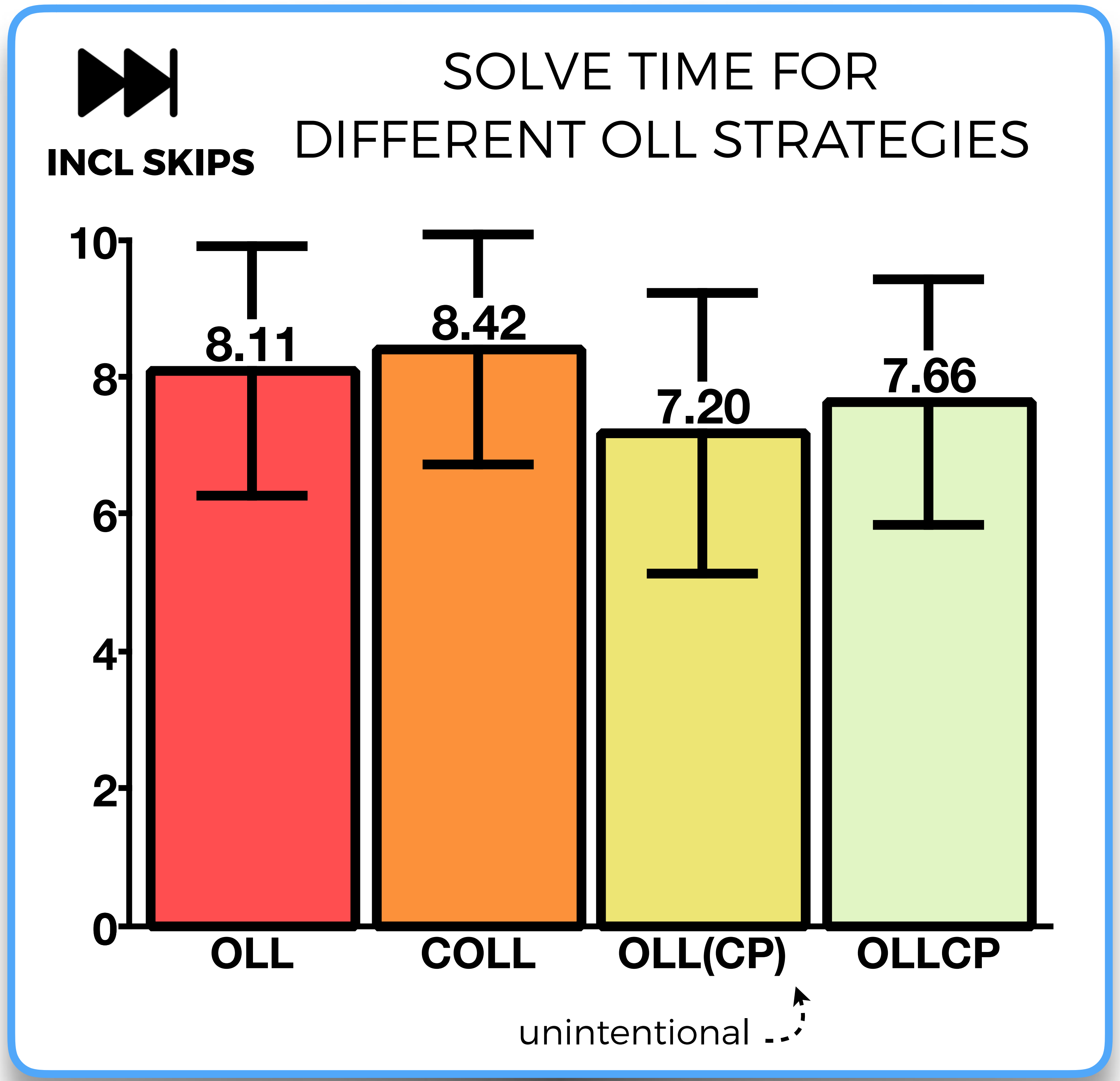
Volatility and risk

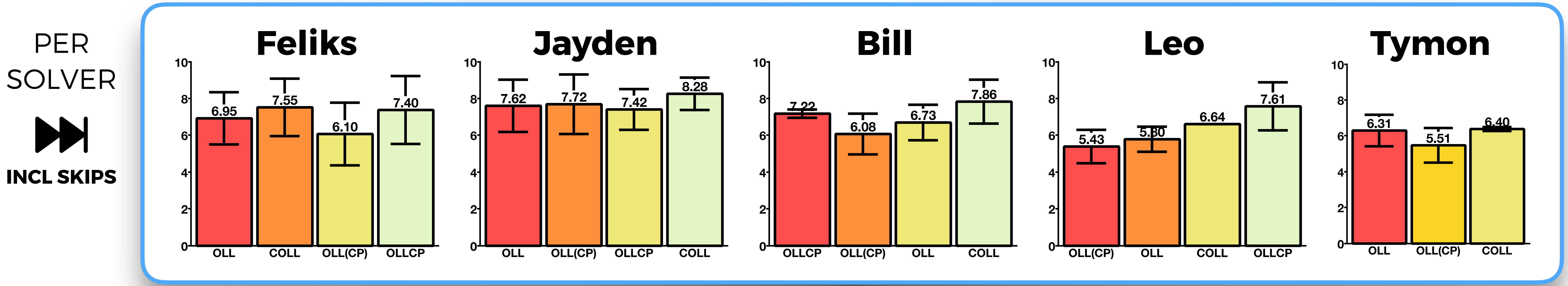
The execution time for several PLL tends to be rather constant (e.g. Na), this makes them less risky than other "faster" PLLs that sometimes are executed very well (e.g. Ua) but other times generate heavy time losses (e.g. T)

N=4000+

OLLCP + EPLL IS AT BEST SIMILAR TO OLL+PLL (A PER-SOLVER ANALYSIS SHOWS OLLCP TO BE USUALLY SLOWER THAN OLL EVEN WHEN SKIPS ARE INCLUDED); COLL DOESN'T SEEM TO BE WORTH IT

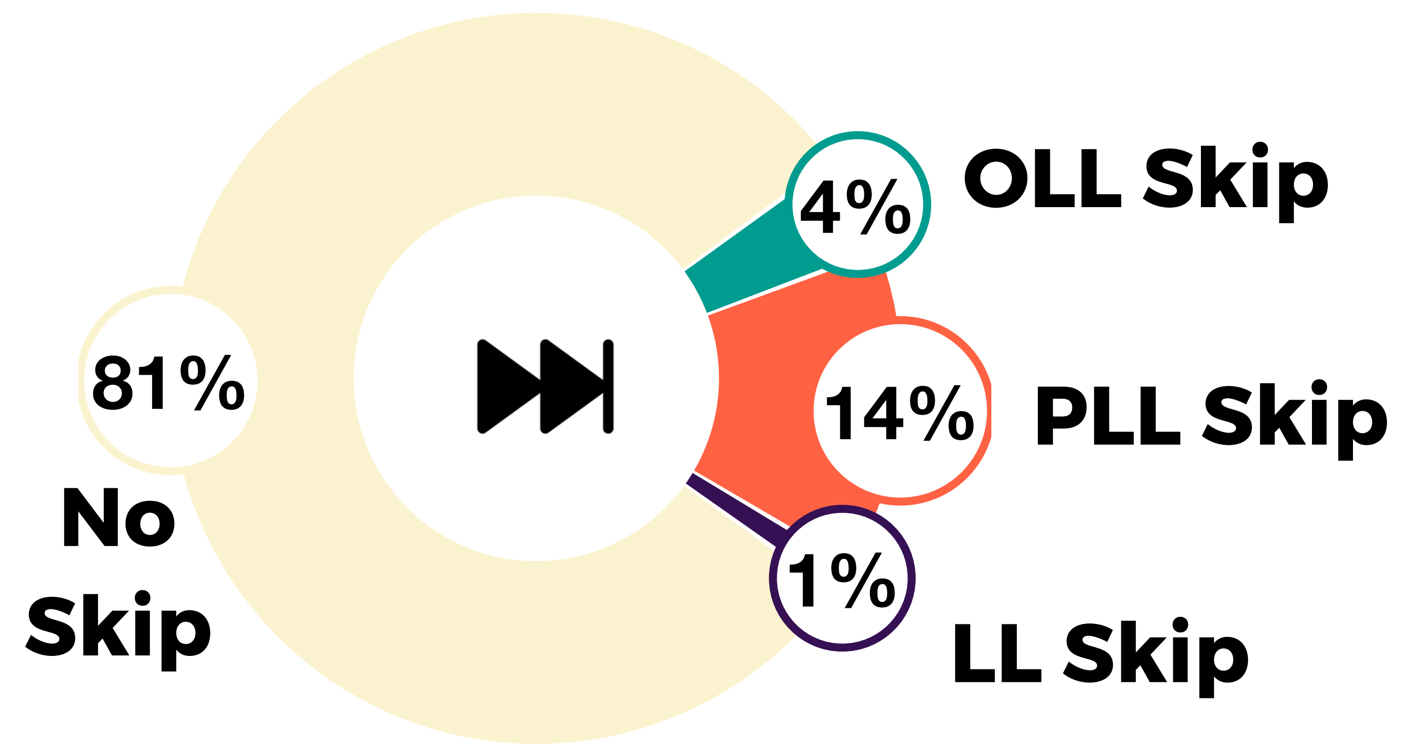




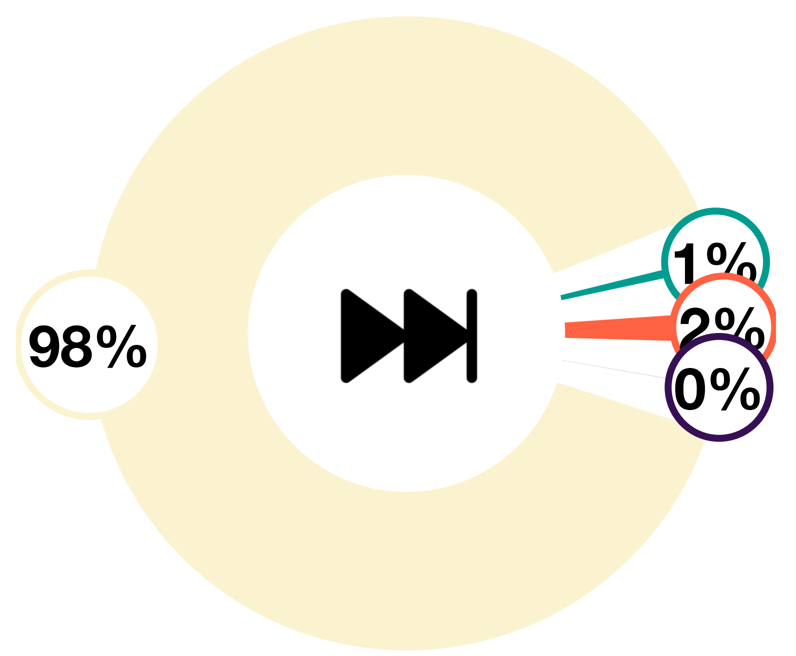


ONE IN FIVE SOLVES ENDS UP IN A SKIPS, WHICH ARE INFLUENCED ALMOST HALF OF THE TIME; A SKIP, ON AVERAGE, SAVES 1.1 SECONDS, WITH PLL BEING VERY SLIGHTLY MORE TIME-SAVING THAN OLL

PREVALENCE OF LAST LAYER SKIPS



NATURAL OCCURRENCES OF SKIPS

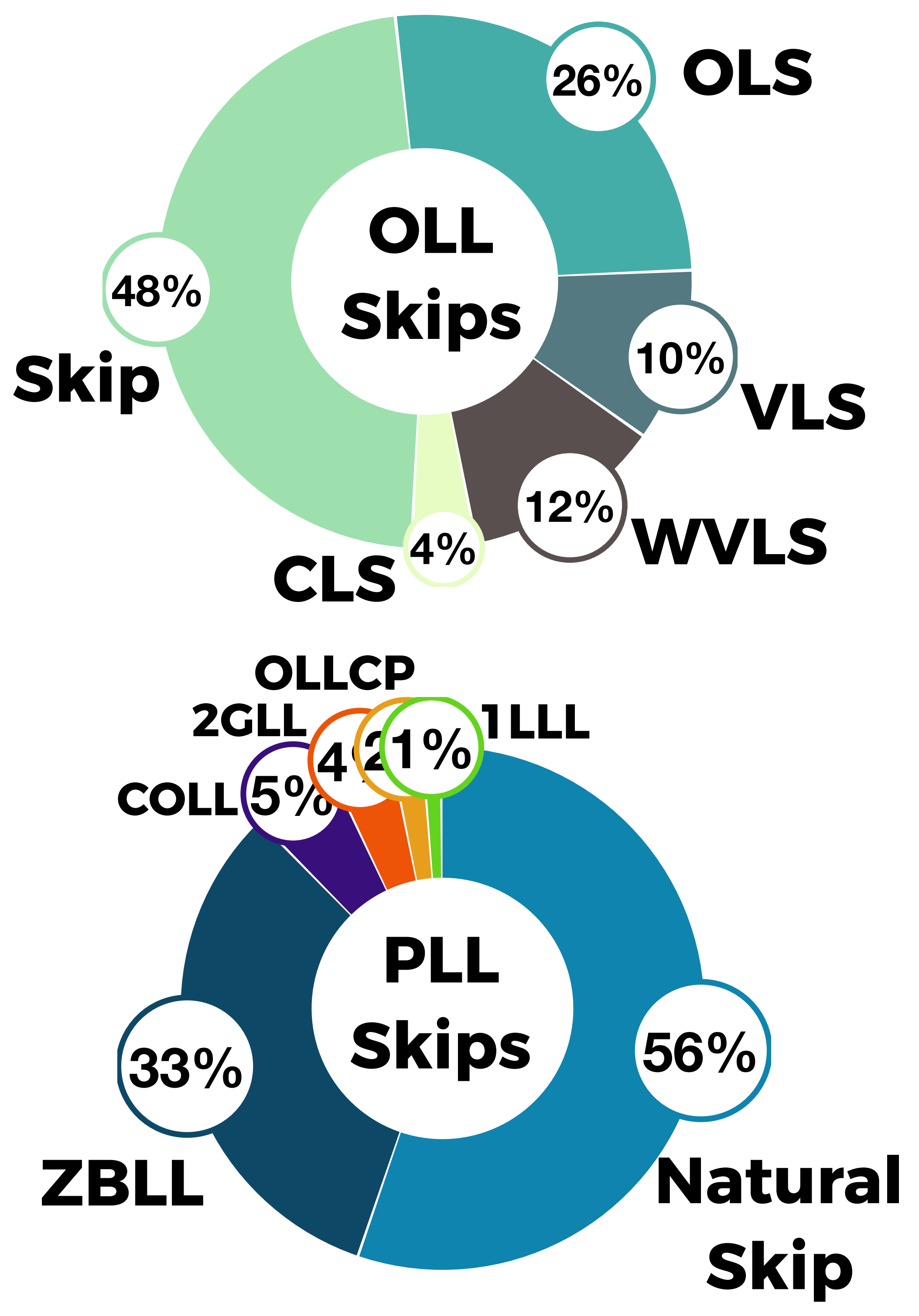


Forced Skip

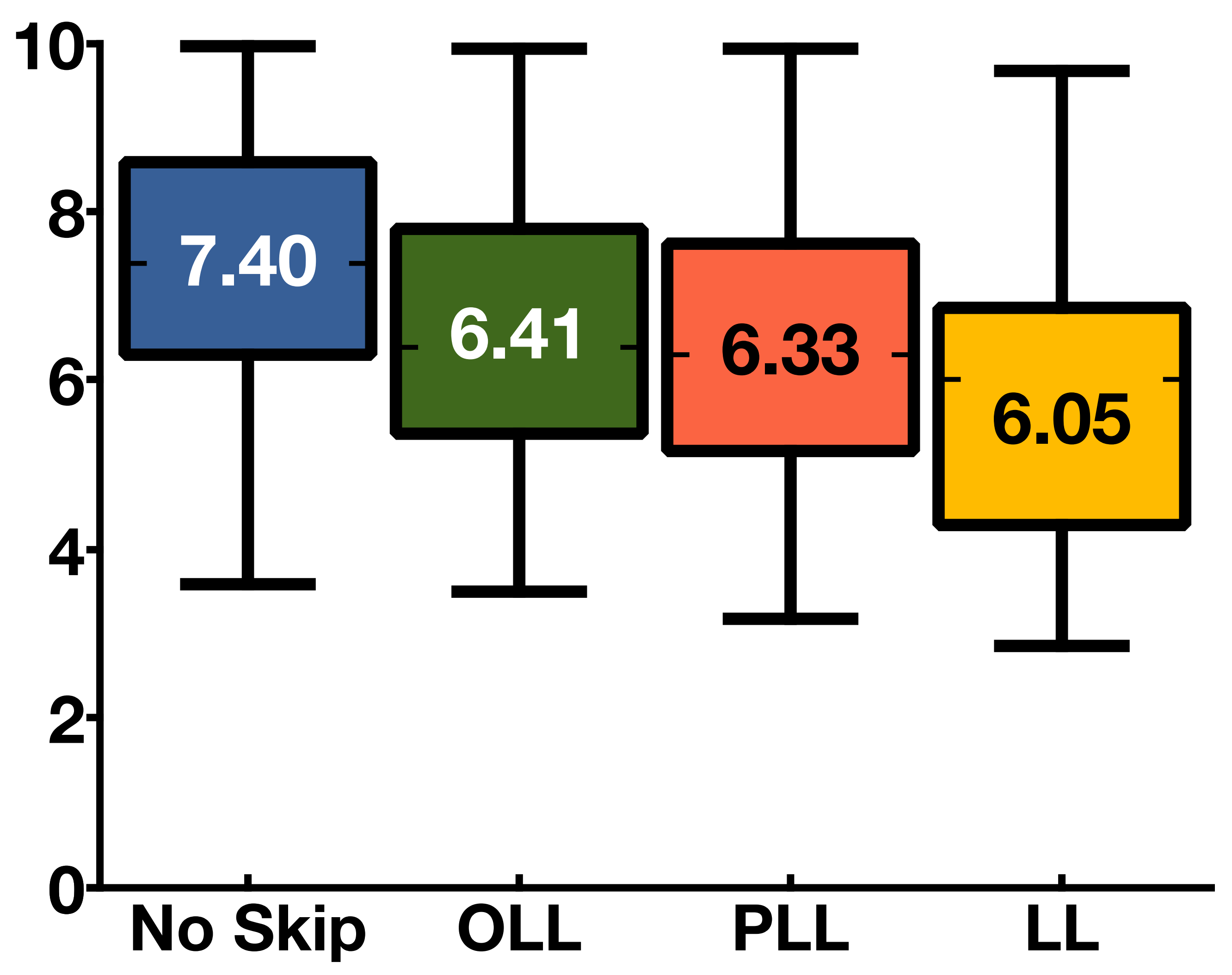
50%

of solves with skip

LAST LAYER INFLUENCING



SOLVE TIME BY TYPE OF SKIP



A biased sample, by necessity

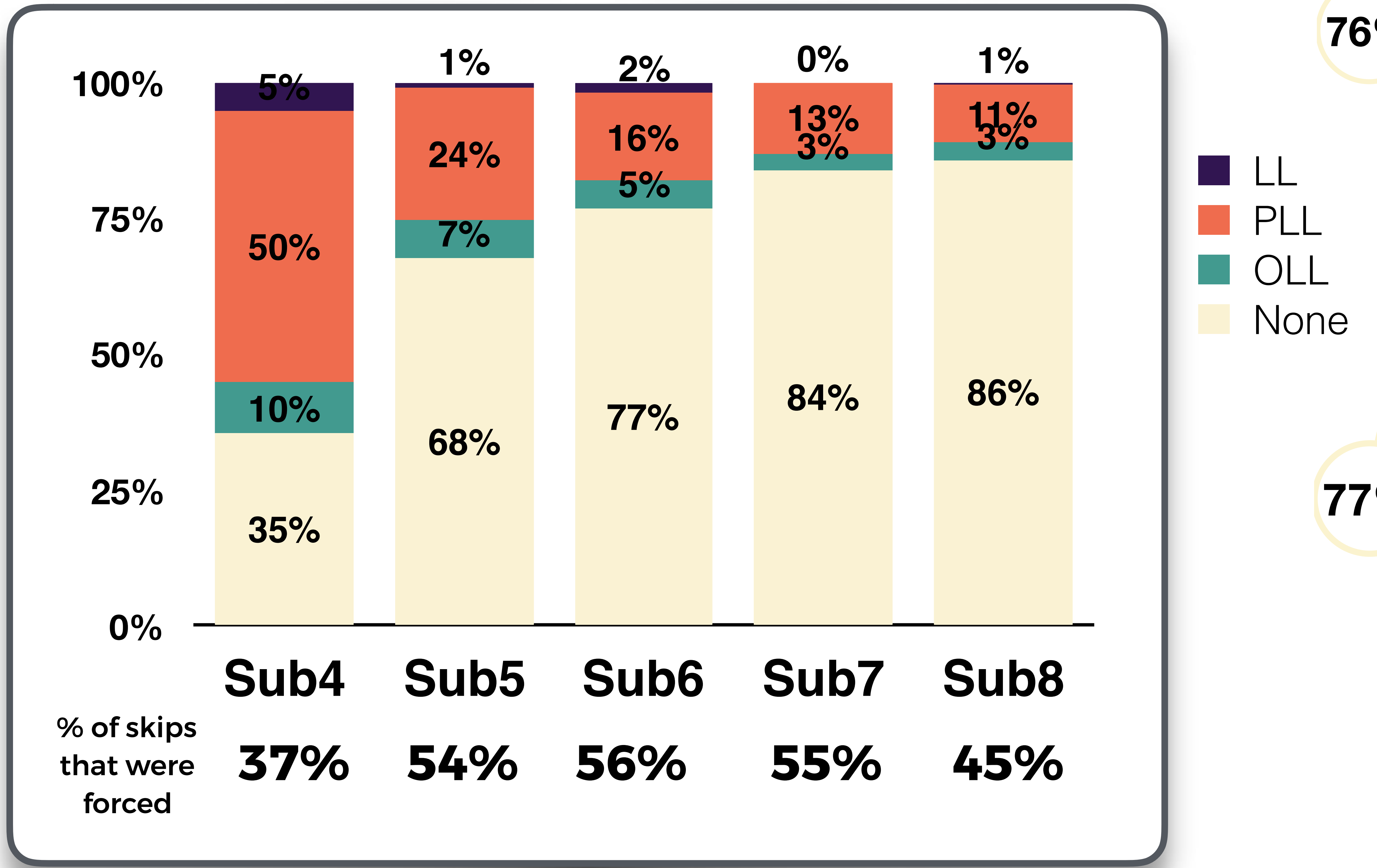
The selection of "good solves" in the dataset, skews the sample significantly compared to a random solve. However, the fact that such a high rate of skips are influenced by the solver tells us that these are important components of the best solves

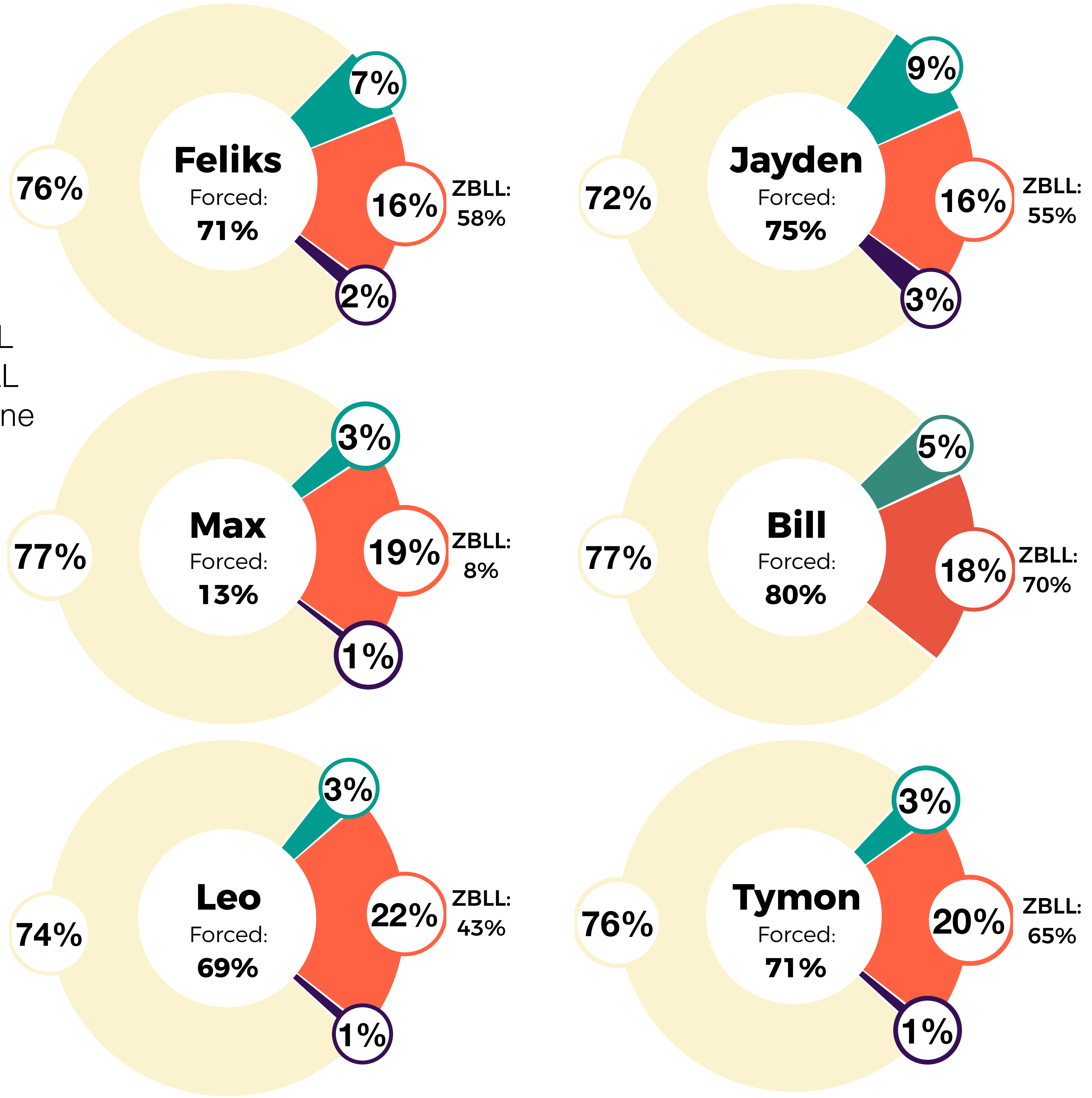
N=4000+

UNSURPRISINGLY, LUCK PLAYS A VERY KEY ROLE FOR THE FASTEST SOLVES; BUT INFLUENCING HAPPENS A LOT, (AND DIFFERENTLY FOR DIFFERENT SOLVERS)

% OF SKIPS BY SOLVE TIME

for <4, 4-5, 5-6, 6-7, 7-8 second solves





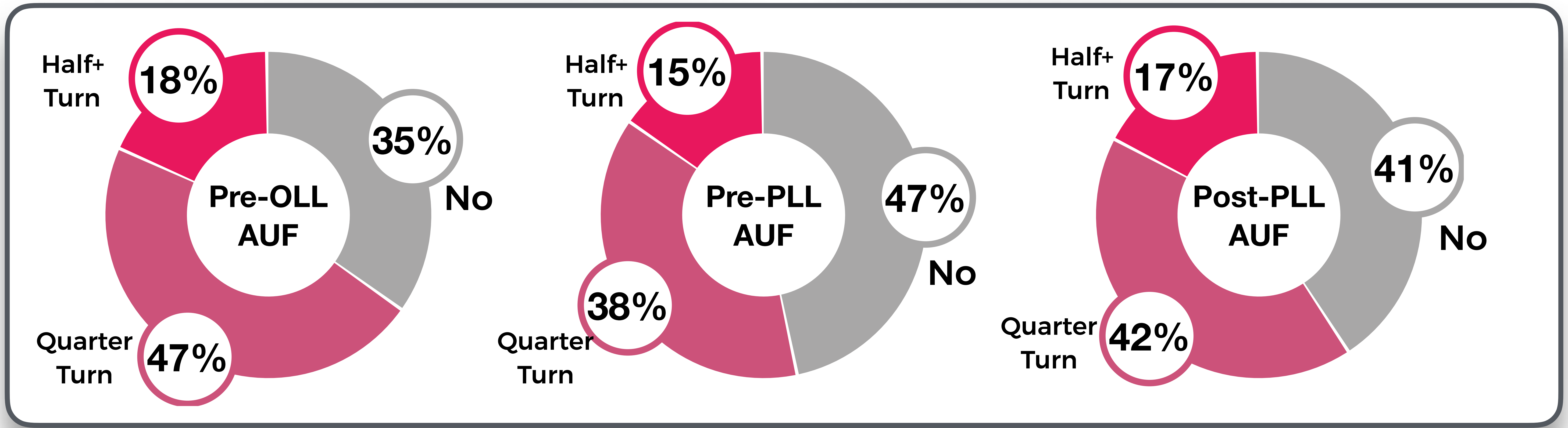
No time to think ?

While the amount of skips is high for the fastest solves, the rate of skip influencing is significantly lower for solves under 4 seconds : is it a process that always slows things down?

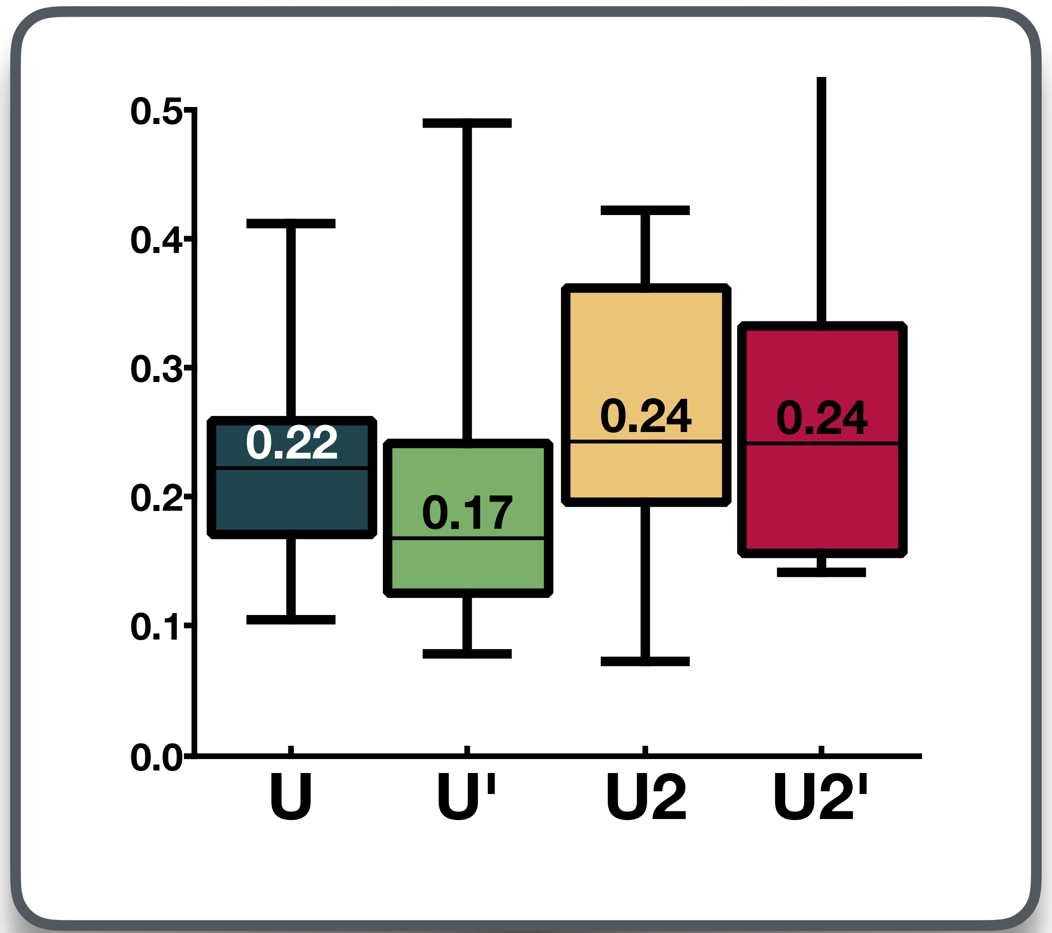
N=4000+

THE MAJORITY OF LAST LAYER ALGS REQUIRE SOME ADJUSTMENT, WITH OLL REQUIRING THE MOST: SOLVERS LEARN ALTERNATIVE ALGS FOR MULTIPLE PLL ANGLES

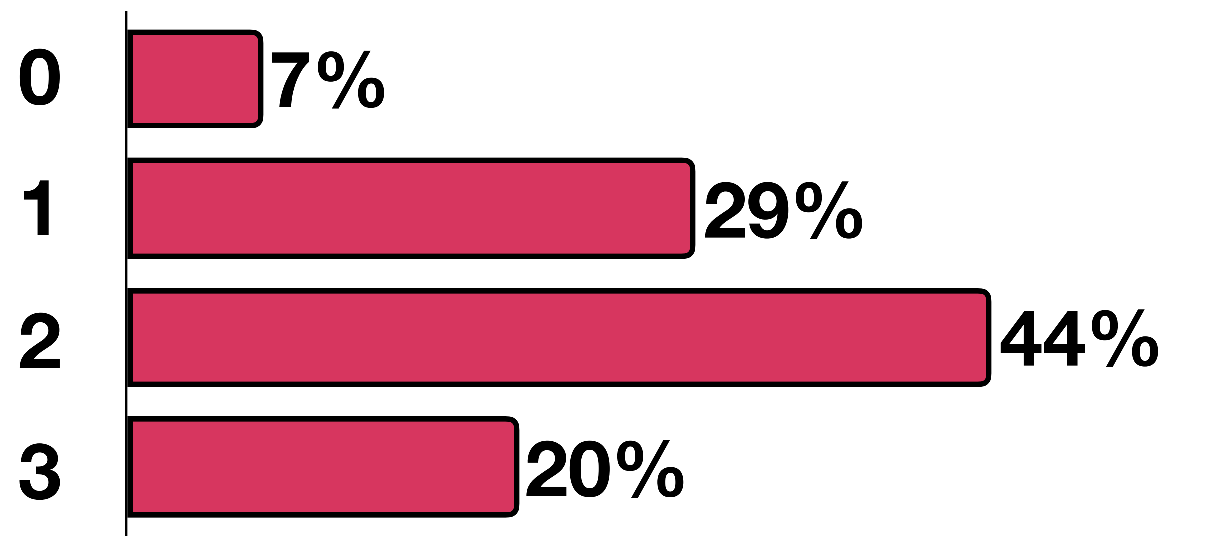
AUF BEFORE AND AFTER EXECUTION OF LAST LAYER ALGS



AUF EXECUTION TIME

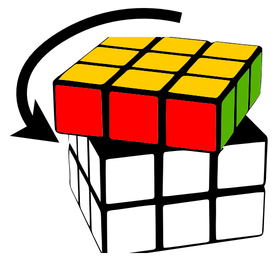


OF AUFS IN LAST LAYER



Death by a thousand cuts

At the level of the fastest solve, a 0.2 sec loss due to AUFs still accounts for 5-7% of the entire solve. not an entirely negligible question therefore whether to strategise around AUF when choosing an alg

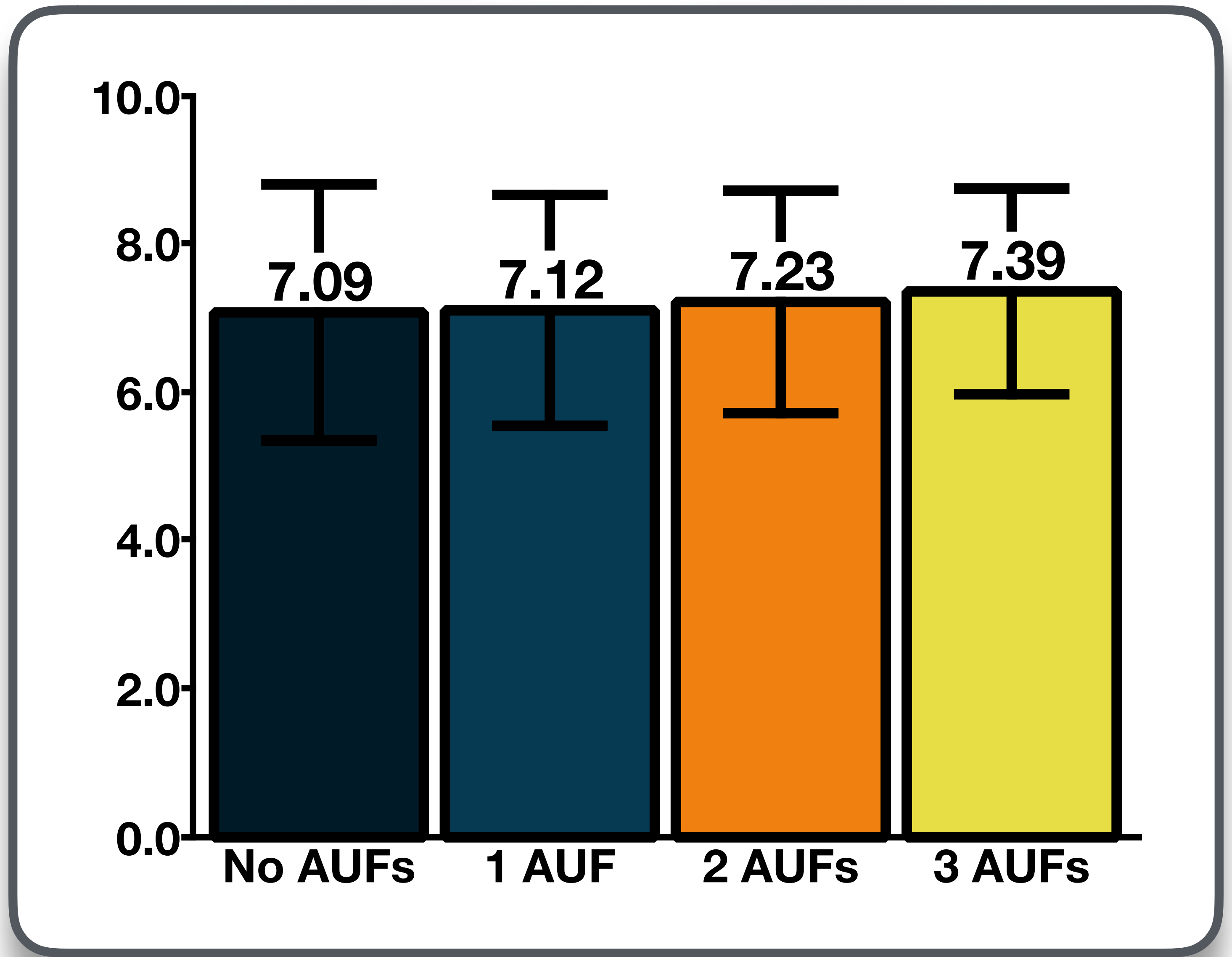


Average LL:
1.77 AUFs

N=4000+

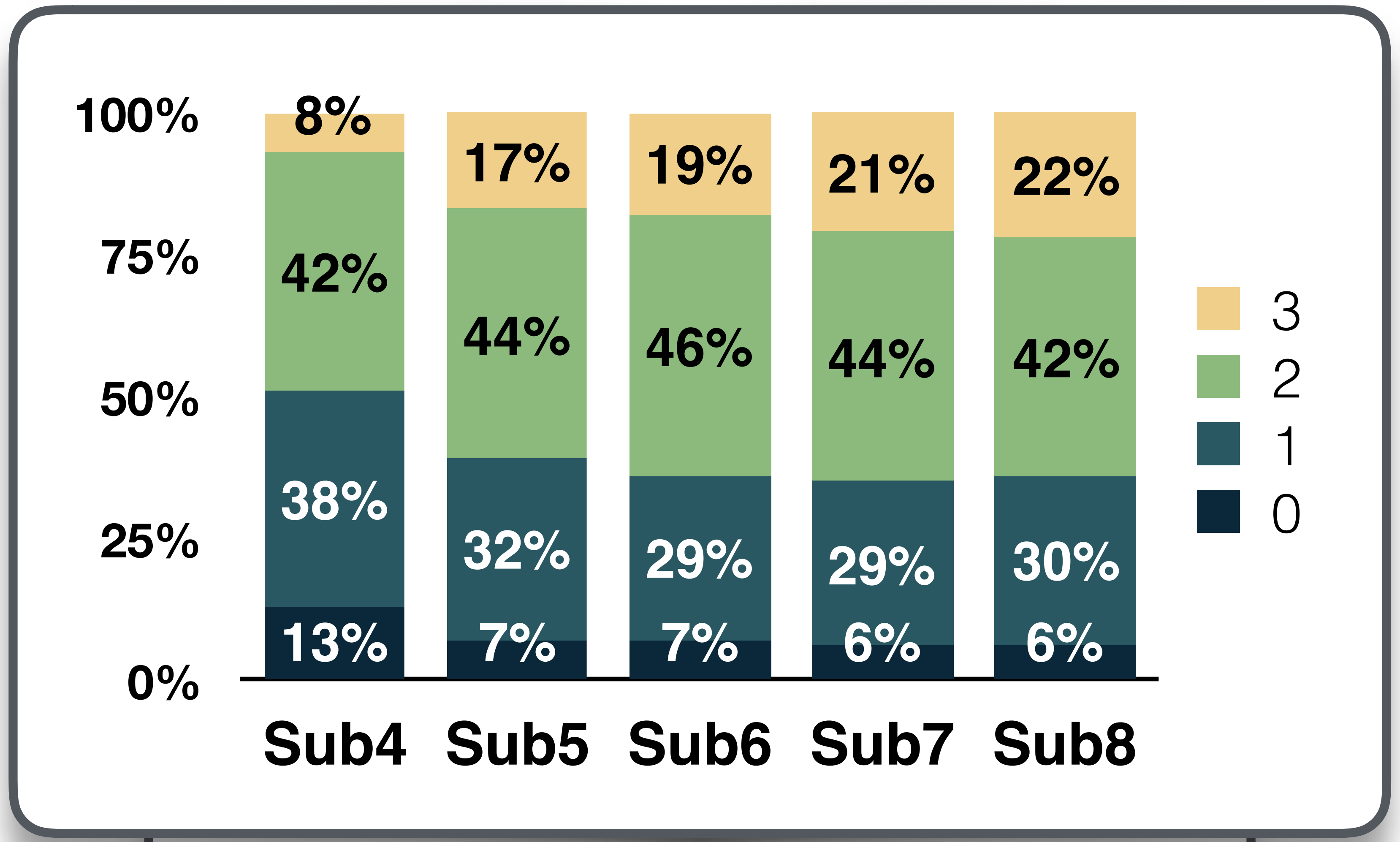
EVERY BIT HELPS FOR THE FASTEST SOLVES : NO-AUFS ARE TWICE AS LIKELY TO OCCUR IN SUB4 SOLVES

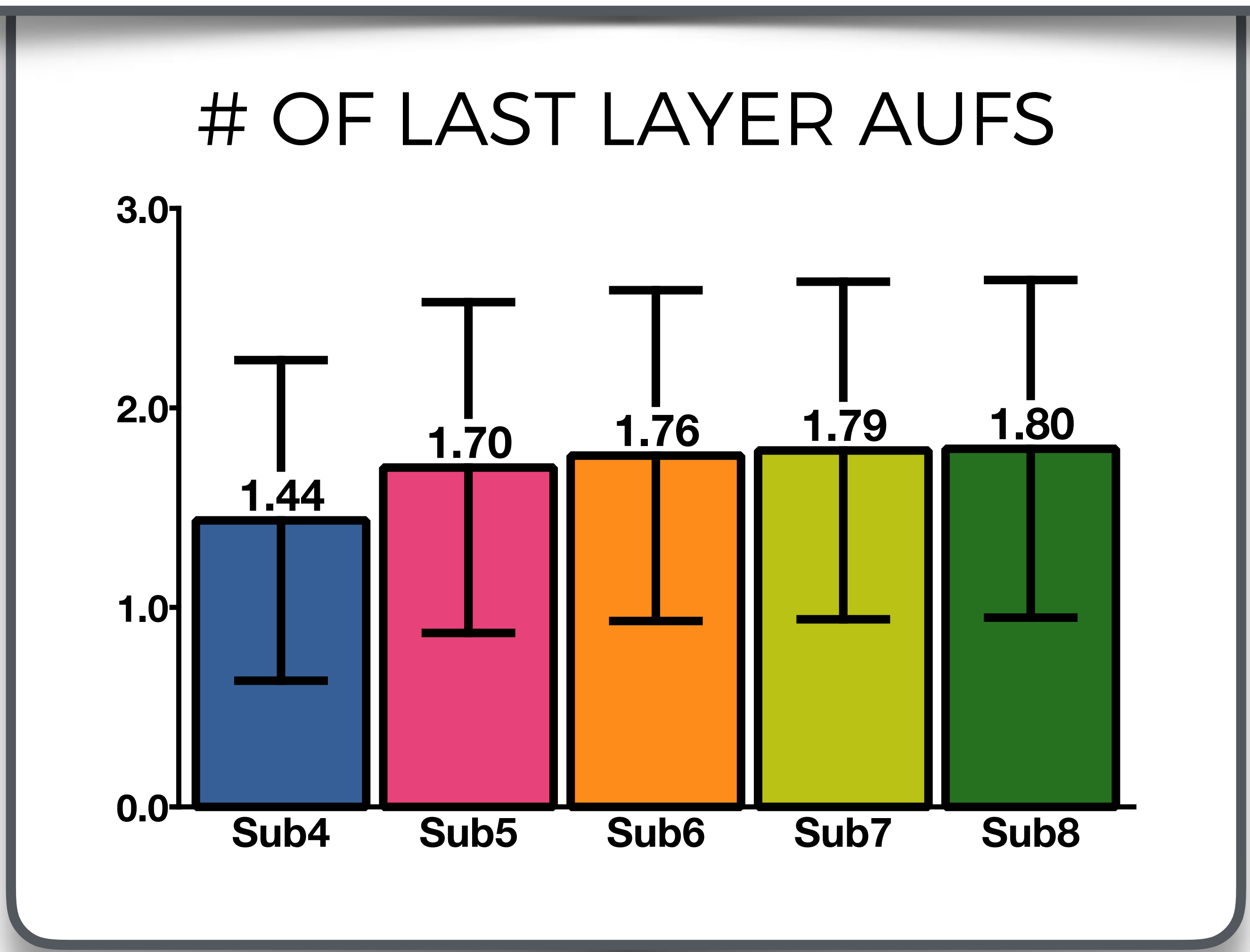
SOLVE TIMES WITH AND WITHOUT LAST LAYER AUFS



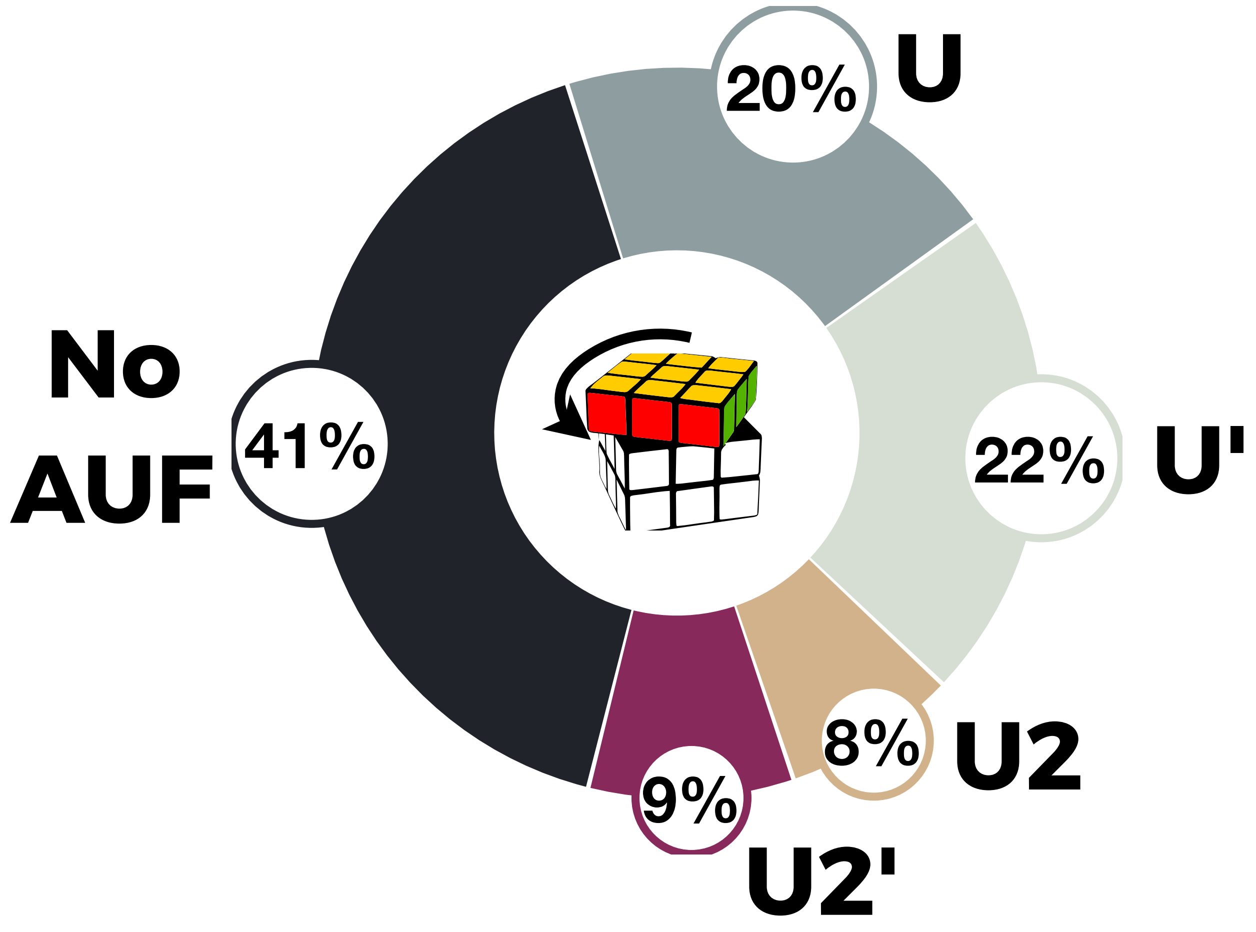
AUFS BY SOLVE TIME

for <4, 4-5, 5-6, 6-7, 7-8 second solves





% OF AUFS AT END OF SOLVE



N=4000+

CONCLUSIONS AND LEARNINGS

WHAT CAN WE LEARN FROM ALL OF THIS

SOME MORE AND SOME LESS SURPRISING FACTS

- **At the fastest speeds, there is a tradeoff between TPS and move efficiency**
  - It might not be possible to be efficient if things are moving too fast
- **The "canon" split for CFOP steps sits at around 16% | 45% | 17% | 22%**
  - For the fastest solves, last layer shrinks (skips), and cross goes up (x(x)crosses)
- **X- and XX-crosses become a necessity for most of the fastest solves**
  - They appear in half of sub4 solves and appear in ~20% of solves on average
- **The vast majority of time standard RUR'-like inserts are good enough**
  - It's an even ~50/50 between joint and split pairs
  - Sledge inserts are very rare (a bit more frequent for last pair, at 6%)
- **Slice moves are a bad idea during F2L, f-move inserts are quite good though**
  - S moves are quite good in OLL, but not so much anywhere else, although that might be because we don't have good algs yet!
- **Never rotate for cross, always rotate for f2l**
  - The time loss due to rotation is important in cross but negligible in f2l, and more than compensated by the gain in speed by keeping the moveset simpler
- **Last Layer skips happen 20% of the time, and solvers are influencing them ~50% of the time**
  - But the fastest solves have a lot fewer forced skips: it is probably time consuming to think about them at those speeds. Is it better to simply play and pray?
- **AUFs are needed 60% of the time**
  - AUFs are less frequent in the faster solves, with that extra bit of luck contributing to the overall "shaving time bit by bit" trend that seems to describe in general the fastest solves

"Keep it simple" seems to be the winning strategy for the fastest CFOP solves

WHAT'S NEXT FOR THIS ANALYSIS

- **The other methods**
  - A recent spurt of efforts has been made into recording and reconstructing Roux solves. Despite this, the data available is still limited. The next challenge is to integrate the existing data into the scab and then conduct a similar analysis on the second of the Big 2
  - Other traditional methods (ZZ, Petrus) have not seen a lot of usage, despite its coterie of stalwart defenders. While I suspect that a large-scale analysis such as the one we present here and the one planned for Roux will not be feasible, many things can still be learnt about these methods
  - Much more recent methods (Mehta), somewhat boutique (zipper) or meme-but-not-only methods (Belt!) can present nuances in solving strategies that might be interesting. A number of awesome people have already or are in the process of contributing sizeable amounts of solves for these methods, so the only barrier left is to put together the analysis itself!
- **Further analysis**
  - Currently all solves are taken together, but given the prevalence of low-solve-count solvers in the data, many KPIs are not encapsulating the variance within solver, and the number of solvers for which we have sufficient solves is (for now) relatively low. This is definitely one area where we'll be working to improve the analysis!
  - Analysis of specific steps in the methods (e.g. F2L inserts, choice of PLL all for specific cases) is for now surface level. Getting smarter tools to identify patterns in the solves and how they influence the outcome is likely to prove a challenging but rewarding endeavour