Stockfish

Un logiciel coopératif d'échecs

Stéphane Nicolet – Université Paris 2

Plan

- Historique
- Une ferme de calcul pour les tests
- Programmation collaborative
- Parallélisme
- Quelques améliorations récentes
- … et maintenant ?

Historique

1770

1965

1978

1997

2008

2017

Vingt ans après

Deep Blue (1997)

- hardware spécialisé, massivement parallèle
- 200 millions de positions par seconde
- anticipe 20 coups en avance
- contre Kasparov : perd 2-4, gagne 3,5-2,5 dans la revanche

Stockfish (2017)

- programme C++ ouvert, tourne sur tous les ordinateurs
- 2 millions de positions par sec. en monothread, 64 threads…
- anticipe 40 coups en avance
- *beaucoup* plus fort que tous les humains

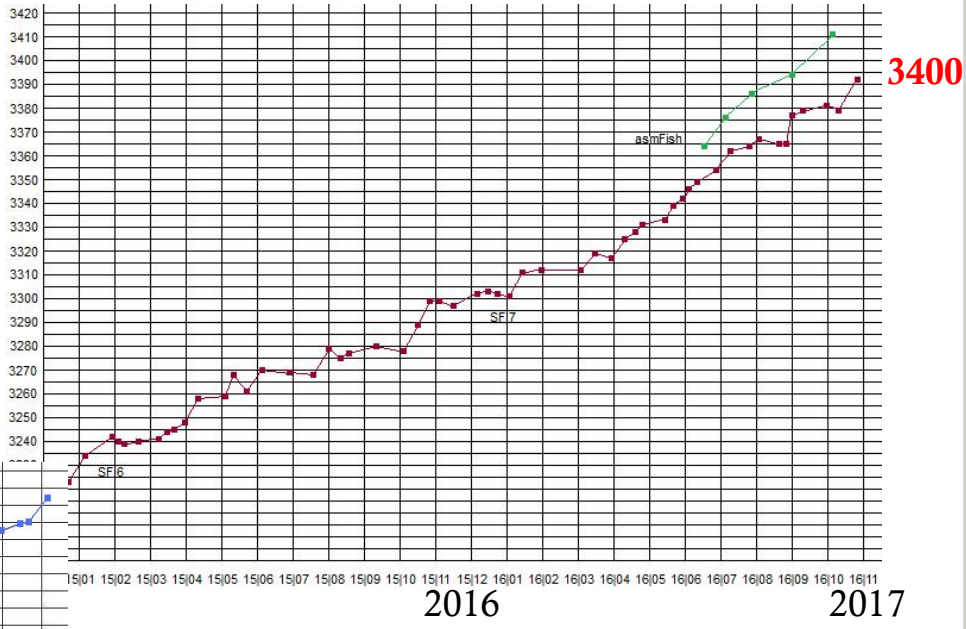
Force brute ou intelligence ?

- Recherche = arbre de jeu + fonction d'évaluation
- Dans les livres : f(P) = combinaison linéaire de 20 termes échiquéens…
- Deep Blue avait environ 100 termes
- Mais dans Stockfish : 850 termes stratégiques + 50 termes de sélectivité. Non linéarité.

- Pas de bibliothèque d'ouvertures, mais bases de finales

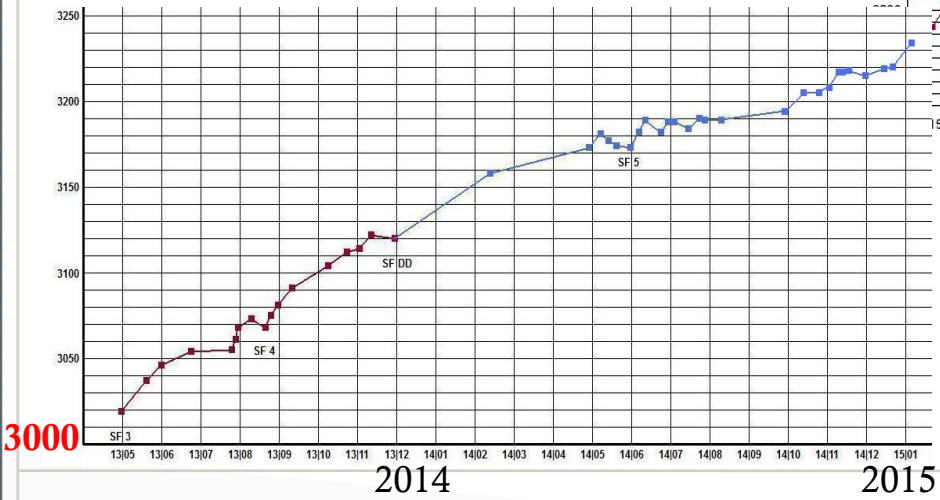
Progression

Stockfish gagne environ 120 points Elo par an



3400

2016 2017



3000

2014 2015

Chaque version est testée et critiquée par la communauté

Rappel : classement Elo

- La probabilité de gain d'un joueur A contre un joueur B est donnée par :

$$p(D) = \frac{1}{1 + 10^{\frac{D}{400}}}$$

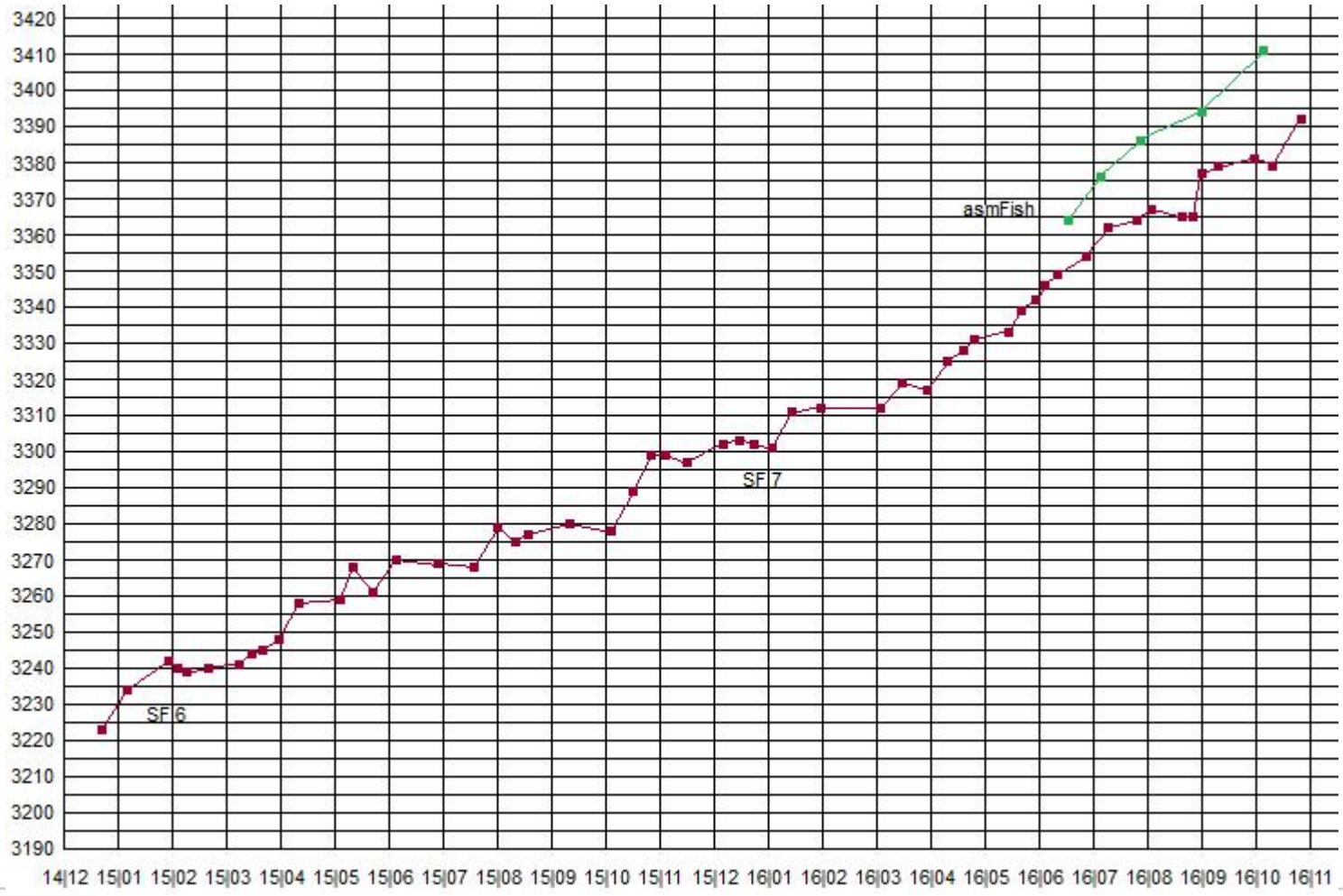
où D est la différence de classements entre B et A

- pour $A = 2830$ et $B = 3400$, on trouve

$$p(D) = 0.036$$

(Carlsen / Stockfish)

Progression (2)

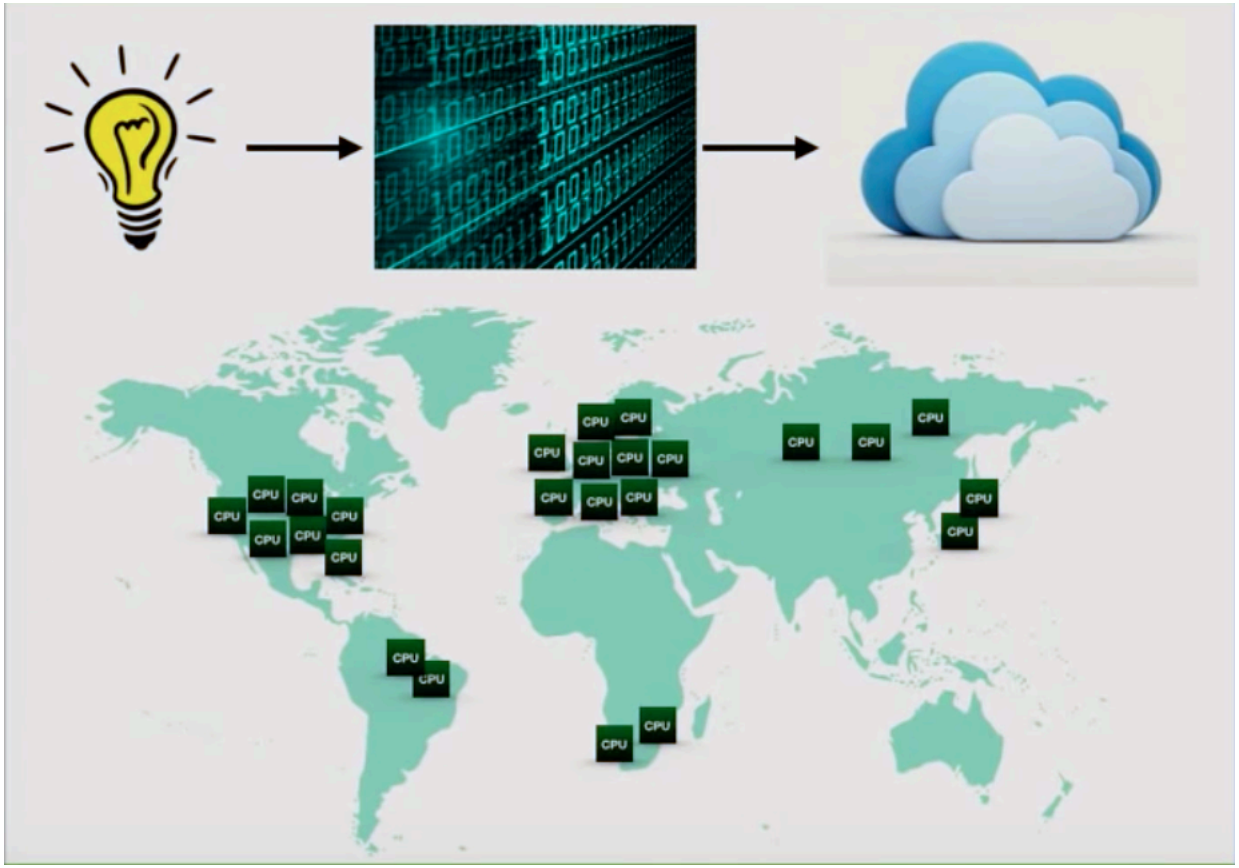


Mode de développement

- Trois auteurs historiques (2008) : Tord Romstad, Marco Costalba, Joona Kiiski
- Chessprogramming wiki, Open-source, GitHub, etc.
- Création de *Fishtest* (2013): Gary Linscott

- Surpasse les programmes commerciaux en 2014
- Une communauté active (100 développeurs)

Fishtest: une ferme de calcul



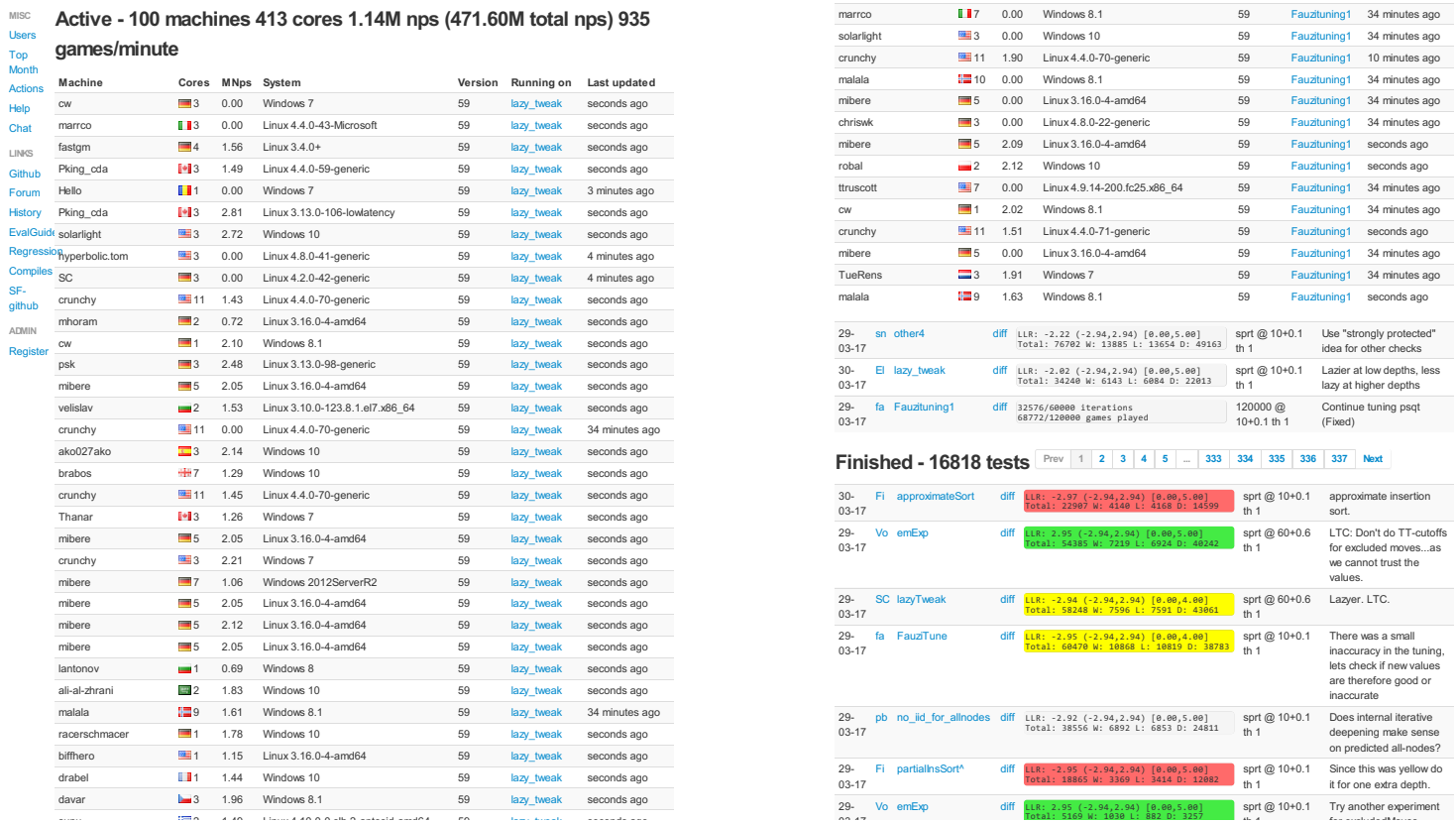
Fishtest (2)

Testers	899	**Active testers**	37	**Games played**	441769135
Developers	116	**Tests submitted**	16721	**CPU time**	683.10 years

- Serveur + clients écrits en Python, disponible
- Testeurs : donnent du temps machine
- Développeurs : codent des modifications

- Chaque modification de la branche master a été testée

Fishtest : le site Web



MISC
Users
Top Month
Actions
Help
Chat

LINKS
Github
Forum
History
EvalGuide
Regression
Compiles
SF-github

ADMIN
Register

Active - 100 machines 413 cores 1.14M nps (471.60M total nps) 935 games/minute

Machine	Cores	MNps	System	Version	Running on	Last updated
cw	3	0.00	Windows 7	59	lazy_tweak	seconds ago
marrco	3	0.00	Linux 4.4.0-43-Microsoft	59	lazy_tweak	seconds ago
fastgm	4	1.56	Linux 3.4.0+	59	lazy_tweak	seconds ago
Pking_cda	3	1.49	Linux 4.4.0-59-generic	59	lazy_tweak	seconds ago
Hello	1	0.00	Windows 7	59	lazy_tweak	3 minutes ago
Pking_cda	3	2.81	Linux 3.13.0-106-lowlatency	59	lazy_tweak	seconds ago
solarlight	3	2.72	Windows 10	59	lazy_tweak	seconds ago
hyperbolic.tom	3	0.00	Linux 4.8.0-41-generic	59	lazy_tweak	4 minutes ago
SC	3	0.00	Linux 4.2.0-42-generic	59	lazy_tweak	4 minutes ago
crunchy	11	1.43	Linux 4.4.0-70-generic	59	lazy_tweak	seconds ago
mhoram	2	0.72	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
cw	1	2.10	Windows 8.1	59	lazy_tweak	seconds ago
psk	3	2.48	Linux 3.13.0-98-generic	59	lazy_tweak	seconds ago
mibere	5	2.05	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
velislav	2	1.53	Linux 3.10.0-123.8.1.el7.x86_64	59	lazy_tweak	seconds ago
crunchy	11	0.00	Linux 4.4.0-70-generic	59	lazy_tweak	34 minutes ago
ako027ako	3	2.14	Windows 10	59	lazy_tweak	seconds ago
brabos	7	1.29	Windows 10	59	lazy_tweak	seconds ago
crunchy	11	1.45	Linux 4.4.0-70-generic	59	lazy_tweak	seconds ago
Thanar	3	1.26	Windows 7	59	lazy_tweak	seconds ago
mibere	5	2.05	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
crunchy	3	2.21	Windows 7	59	lazy_tweak	seconds ago
mibere	7	1.06	Windows 2012ServerR2	59	lazy_tweak	seconds ago
mibere	5	2.05	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
mibere	5	2.12	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
mibere	5	2.05	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
lantonov	1	0.69	Windows 8	59	lazy_tweak	seconds ago
ali-al-zhrani	2	1.83	Windows 10	59	lazy_tweak	seconds ago
malala	9	1.61	Windows 8.1	59	lazy_tweak	34 minutes ago
racerschmacer	1	1.78	Windows 10	59	lazy_tweak	seconds ago
biffhero	1	1.15	Linux 3.16.0-4-amd64	59	lazy_tweak	seconds ago
drabel	1	1.44	Windows 10	59	lazy_tweak	seconds ago
davar	3	1.96	Windows 8.1	59	lazy_tweak	seconds ago

marrco	7	0.00	Windows 8.1	59	Fauztuning1	34 minutes ago
solarlight	3	0.00	Windows 10	59	Fauztuning1	34 minutes ago
crunchy	11	1.90	Linux 4.4.0-70-generic	59	Fauztuning1	10 minutes ago
malala	10	0.00	Windows 8.1	59	Fauztuning1	34 minutes ago
mibere	5	0.00	Linux 3.16.0-4-amd64	59	Fauztuning1	34 minutes ago
chriswk	3	0.00	Linux 4.8.0-22-generic	59	Fauztuning1	34 minutes ago
mibere	5	2.09	Linux 3.16.0-4-amd64	59	Fauztuning1	seconds ago
robal	2	2.12	Windows 10	59	Fauztuning1	seconds ago
ttruscott	7	0.00	Linux 4.9.14-200.fc25.x86_64	59	Fauztuning1	34 minutes ago
cw	1	2.02	Windows 8.1	59	Fauztuning1	34 minutes ago
crunchy	11	1.51	Linux 4.4.0-71-generic	59	Fauztuning1	seconds ago
mibere	5	0.00	Linux 3.16.0-4-amd64	59	Fauztuning1	34 minutes ago
TueRens	3	1.91	Windows 7	59	Fauztuning1	34 minutes ago
malala	9	1.63	Windows 8.1	59	Fauztuning1	seconds ago

29-03-17	sn	other4	diff	LLR: -2.22 (-2.94,2.94) [0.00,5.00] Total: 76702 W: 13885 L: 13654 D: 49163	sprt @ 10+0.1 th 1	Use "strongly protected" idea for other checks
30-03-17	El	lazy_tweak	diff	LLR: -2.02 (-2.94,2.94) [0.00,5.00] Total: 34240 W: 6143 L: 6084 D: 22013	sprt @ 10+0.1 th 1	Lazier at low depths, less lazy at higher depths
29-03-17	fa	Fauztuning1	diff	32576/60000 iterations 68772/120000 games played	120000 @ 10+0.1 th 1	Continue tuning psqt (Fixed)

Finished - 16818 tests Prev 1 2 3 4 5 ... 333 334 335 336 337 Next

30-03-17	Fi	approximateSort	diff	LLR: -2.97 (-2.94,2.94) [0.00,5.00] Total: 22907 W: 4140 L: 4168 D: 14599	sprt @ 10+0.1 th 1	approximate insertion sort.
29-03-17	Vo	emExp	diff	LLR: 2.95 (-2.94,2.94) [0.00,5.00] Total: 54385 W: 7219 L: 6924 D: 40242	sprt @ 60+0.6 th 1	LTC: Don't do TT-cutoffs for excluded moves...as we cannot trust the values.
29-03-17	SC	lazyTweak	diff	LLR: -2.94 (-2.94,2.94) [0.00,4.00] Total: 58248 W: 7596 L: 7591 D: 43061	sprt @ 60+0.6 th 1	Lazyer. LTC.
29-03-17	fa	FauziTune	diff	LLR: -2.95 (-2.94,2.94) [0.00,4.00] Total: 60470 W: 10868 L: 10819 D: 38783	sprt @ 10+0.1 th 1	There was a small inaccuracy in the tuning, lets check if new values are therefore good or inaccurate
29-03-17	pb	no_iid_for_allnodes	diff	LLR: -2.92 (-2.94,2.94) [0.00,5.00] Total: 38556 W: 6892 L: 6853 D: 24811	sprt @ 10+0.1 th 1	Does internal iterative deepening make sense on predicted all-nodes?
29-03-17	Fi	partialInsSort^	diff	LLR: -2.95 (-2.94,2.94) [0.00,5.00] Total: 18865 W: 3369 L: 3414 D: 12082	sprt @ 10+0.1 th 1	Since this was yellow do it for one extra depth.
29-03-17	Vo	emExp	diff	LLR: 2.95 (-2.94,2.94) [0.00,5.00] Total: 5169 W: 1030 L: 882 D: 3257	sprt @ 10+0.1 th 1	Try another experiment

Pull request

- Deux types : simplification ou amélioration
- Toujours le résultat d'un double test réussi
- Discussion de la qualité du code, du ratio intérêt/ complexité, etc…

- Les régulateurs ont le dernier mot

Pull request : exemple

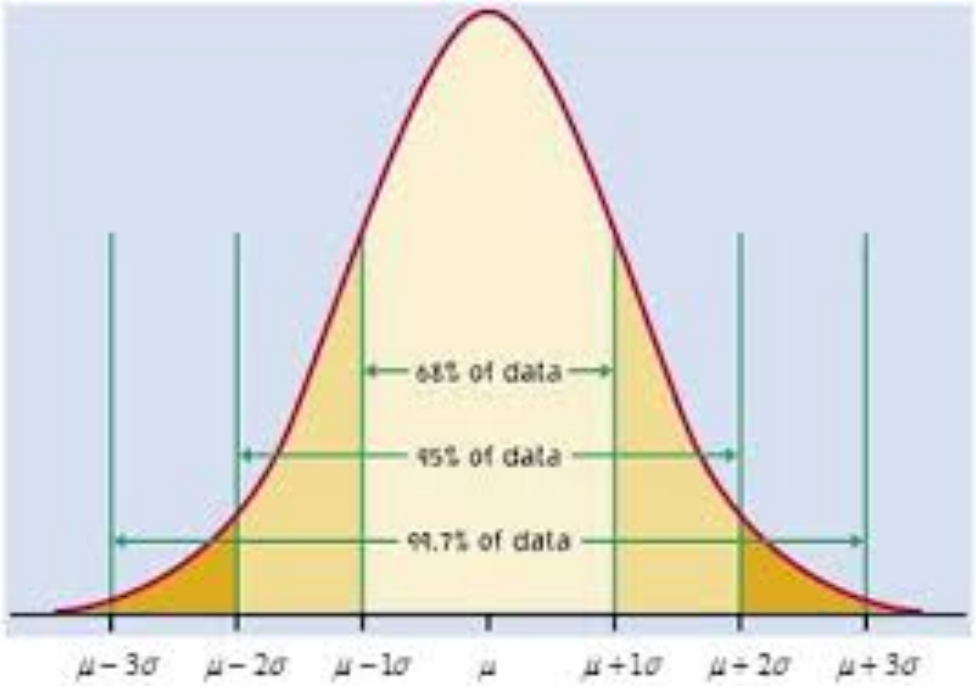
Cf démo sur les pages web…

http://tests.stockfishchess.org/tests

http://github.com/official-stockfish/Stockfish/pull/594

Comment tester ?

- Idée brutale : 40000 parties ultra-blitz + 40000 parties blitz



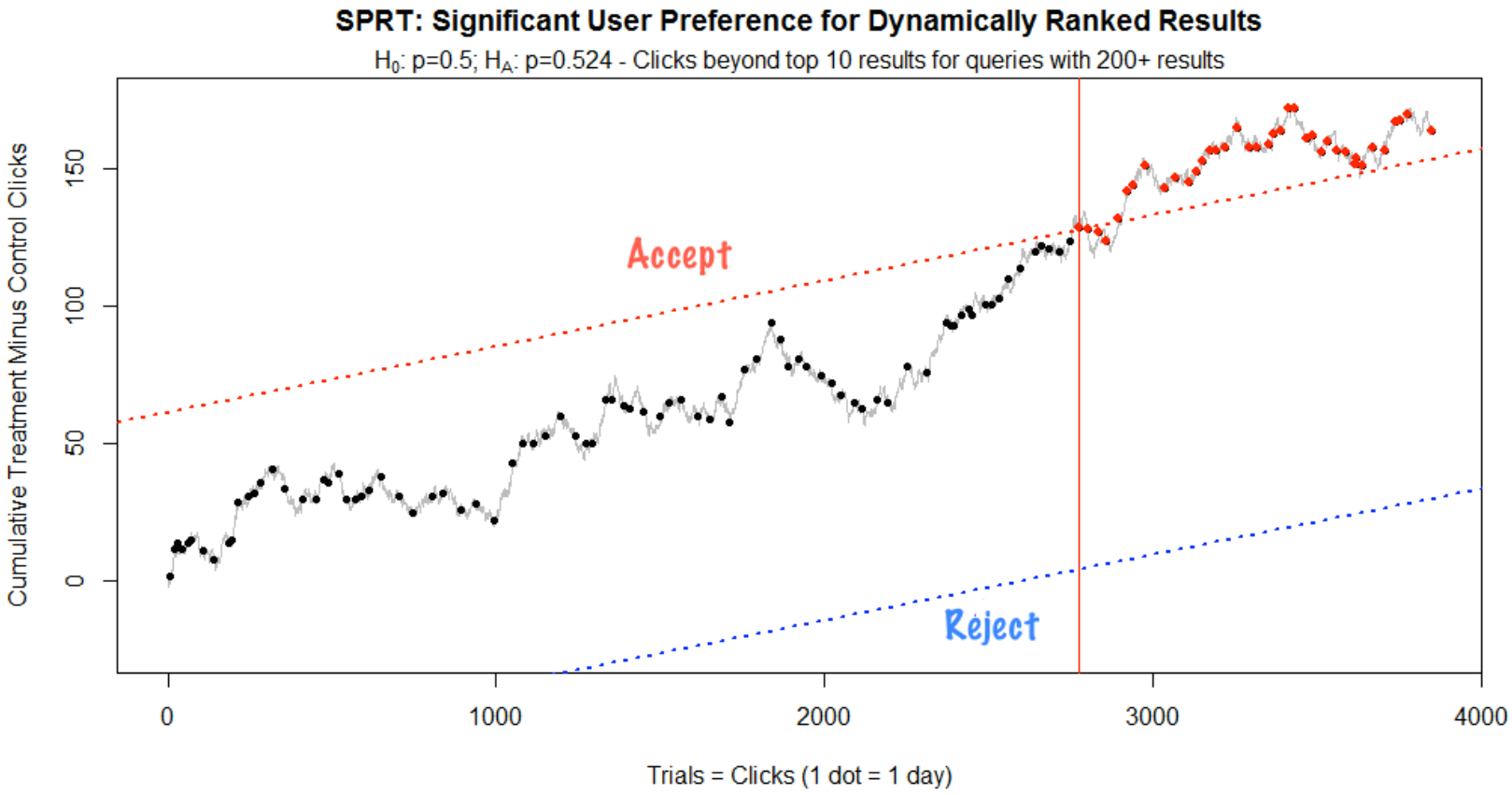
SPRT

- Meilleure idée : Sequential Probability Ratio Test (Wald, 1945)
- SPRT(m..b) : *P(X > b)* et *P(X < m)* dépendent des simulations.
- On pose $r = \frac{P(X > b)}{P(X < m)}$ et on s'arrête dès que

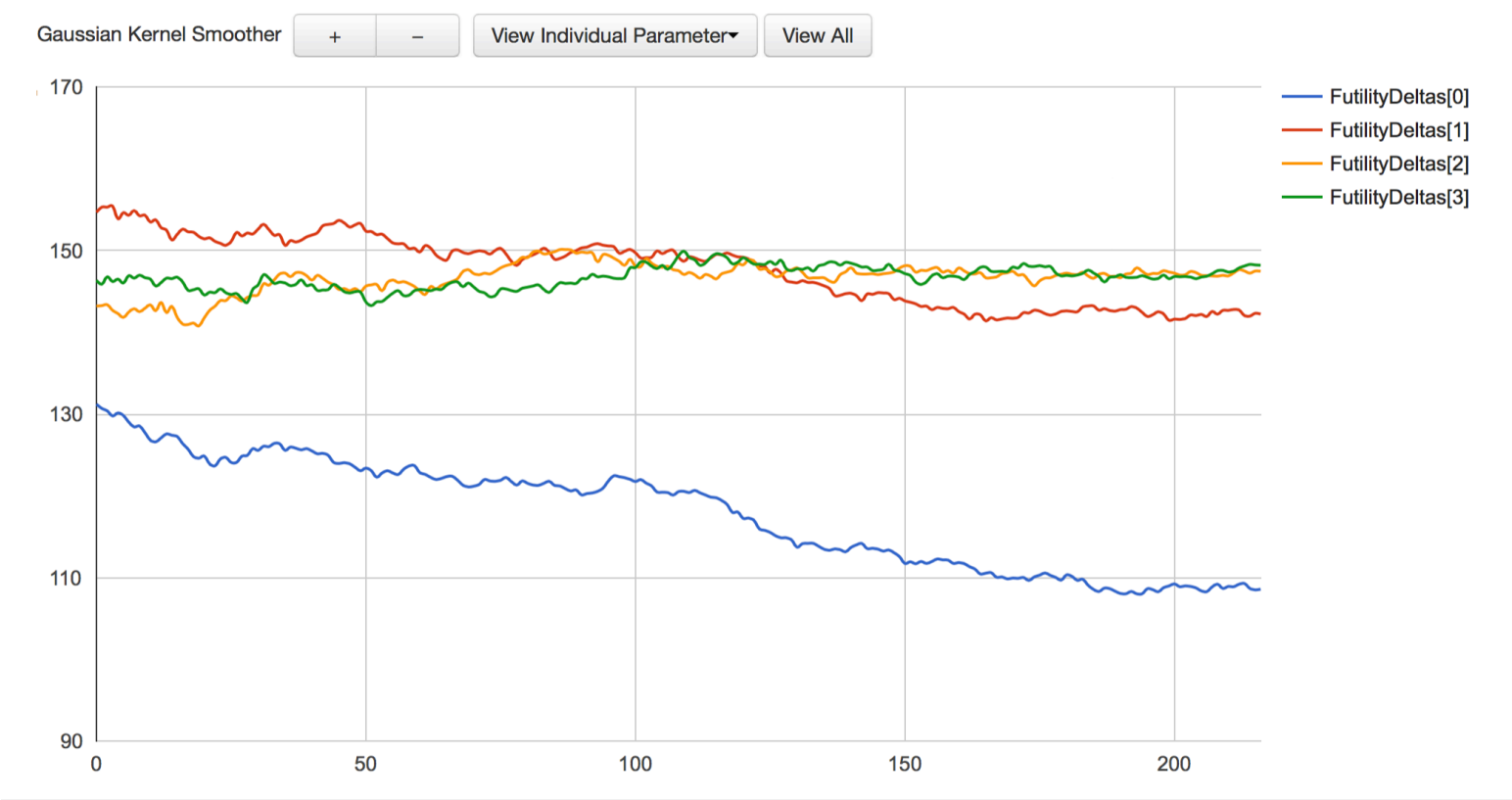
  r > 20 (réussite) ou que r < 1/20 (raté)

- Choix des bornes ? Amélioration : (m..b) = (0..5)
  Simplification : (m..b) = (-3..1)

SPRT (2)

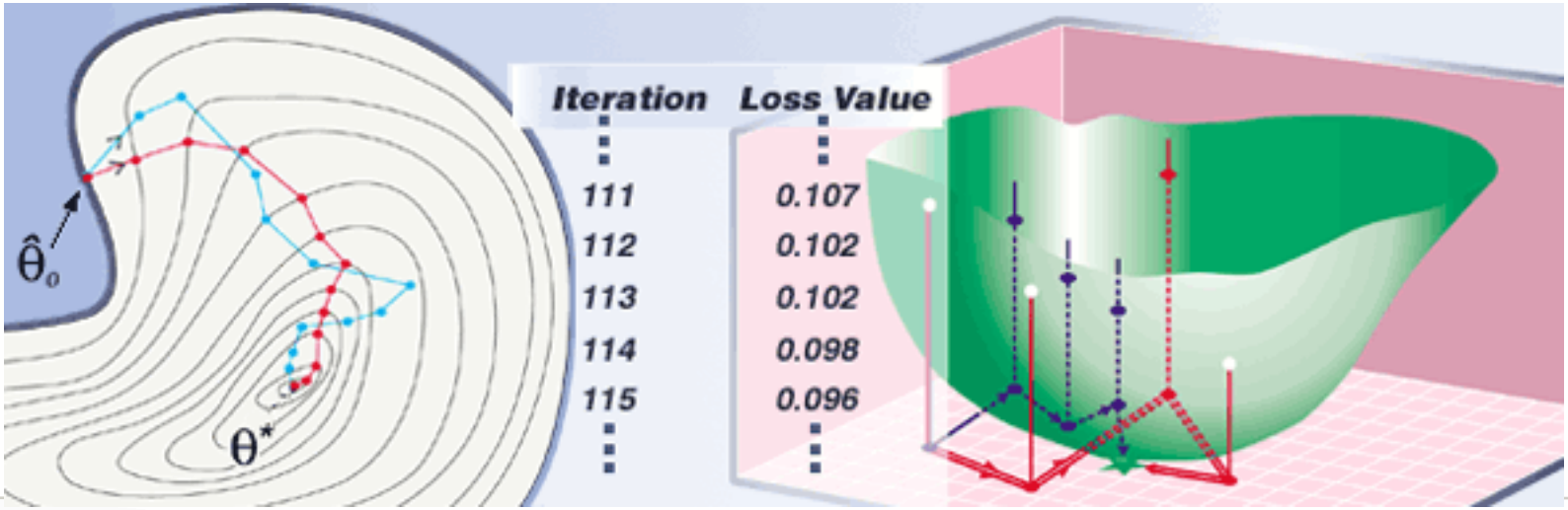

SPRT: Significant User Preference for Dynamically Ranked Results
H0: p=0.5; HA: p=0.524 - Clicks beyond top 10 results for queries with 200+ results
Cumulative Treatment Minus Control Clicks
0
50
100
150
Accept
Reject
0
1000
2000
3000
4000
Trials = Clicks (1 dot = 1 day)

Optimisation des coefficients



Optimisation des coefficients (2)

- Idée 1 : descente de gradient pour minimiser une erreur sur une base de données
- Idée 2 : descente de gradient pour maximiser le score contre la version précédente



Choix du gradient

Classique

- approximation du gradient

$$grad_i(\theta) = \frac{f(\theta + c.e_i) - f(\theta - c.e_i)}{2c}$$

- *2d* évaluations pour estimer le gradient en dimension *d*

SPSA (Spall, 1998)

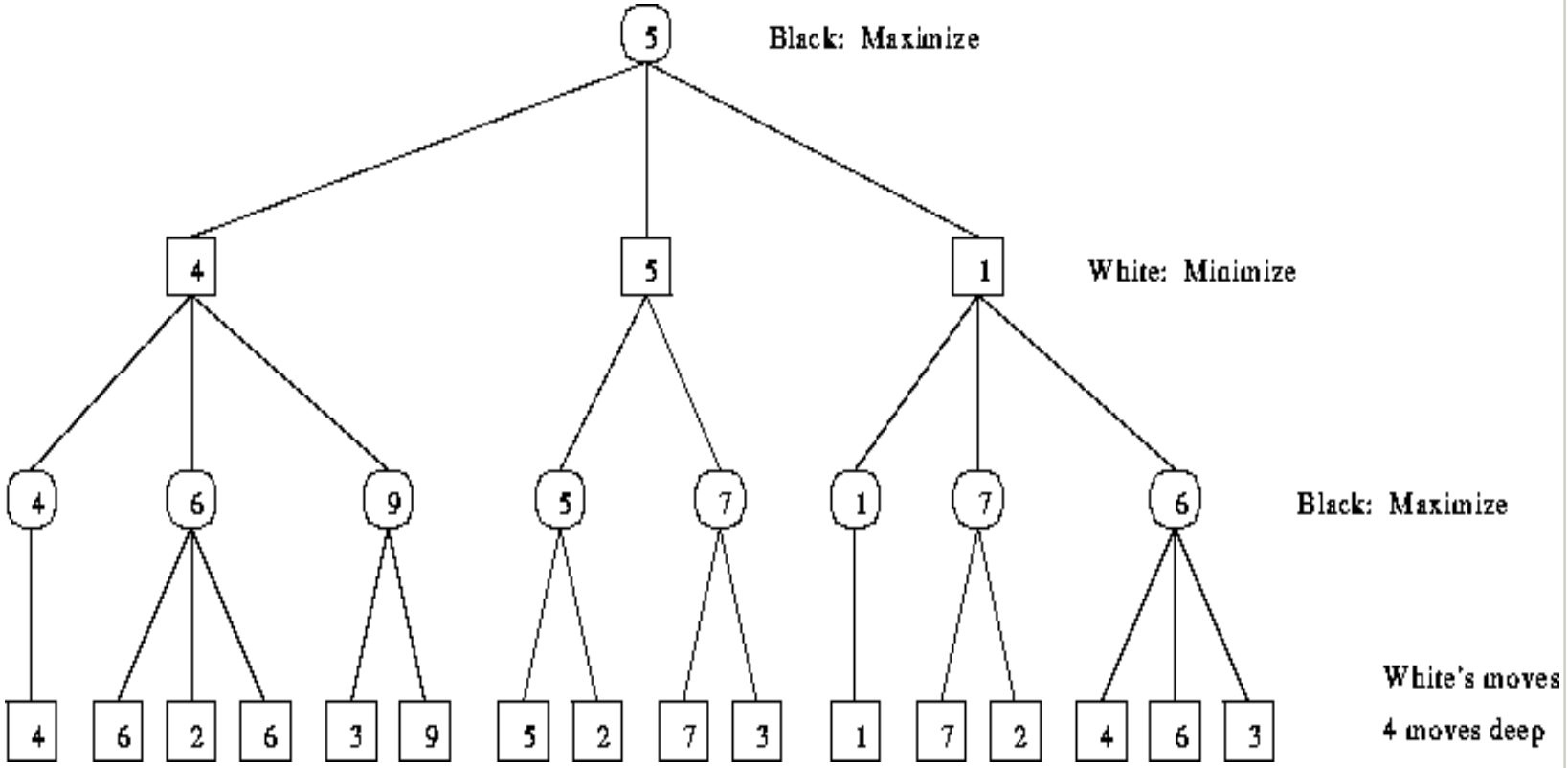
- Simultaneous perturbation stochastic approximation

$$grad_i(\theta) = \frac{f(\theta + c\Delta) - f(\theta - c\Delta)}{2c\Delta_i}$$

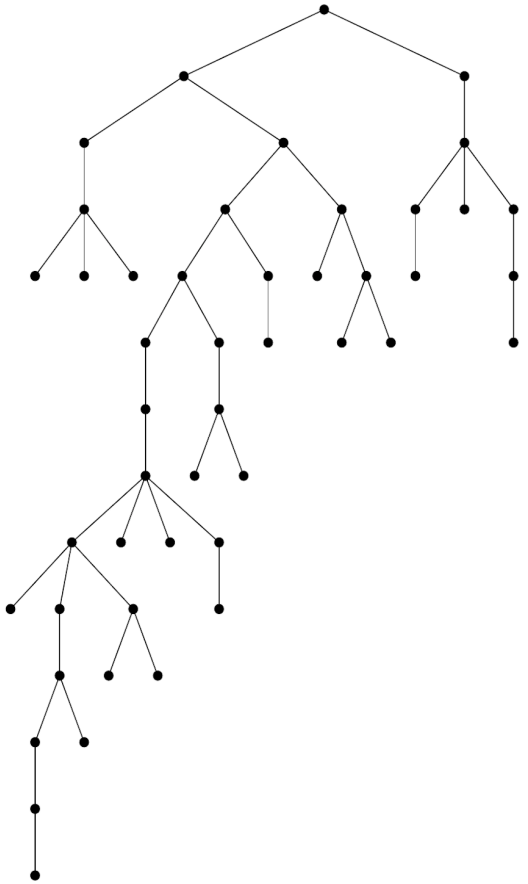
où Δ est un vecteur aléatoire rempli de +1 ou -1

- Seulement *2* évaluations, quelle que soit la dimension *d*

Parallélisation



Parallélisation (2)



Paralléliser alpha-bêta :

• problème intrinsèquement difficile

• facteur de branchement théorique (Knuth, Pearl) :

$$b^{d/2}$$

• aux échecs, b = 35 mais le branchement observé dans Stockfish vaut 2…

• raisons : late move reduction, élagages agressifs, etc

Lazy SMP

- *Idée* : supprimer toutes les primitives de synchronisation
- La communication se fait par la table de transposition
- Les threads auxiliaires sont en avance sur la thread principale

Résultat : +25 Elo à 48 threads

Améliorations récentes

- *Amélioration de l'algorithme de recherche :*

  réutiliser les « plans » plutôt que les coups

- *Amélioration de l'évaluation des positions :*

  gestion du risque à l'intérieur même de la fonction d'évaluation

…et maintenant ?

Pistes de recherche :

- réseaux de neurones ?

- Monte-Carlo ?

- méthode hybride utilisant l'évaluation f(P) pour guider la recherche Monte-Carlo ?