Experimentální analýza algoritmů včera a dnes

1) Typický příklad z roku 1980:

W. Dobosiewicz: An efficient variation of bubble sort

(Information Processing Letters 11 (1980), 5-6)

prezentuje nový algoritmus

teoretickou složitost nespočítá – ta ostatně není spočítaná dodnes

jediný výsledek – B. Brejová: Analyzing variants of Shellsort

(Inform. Proc. Letters 79 (2001), 223-227)

dolní odhad očekávané složitosti je $\Omega\left(\frac{n^2}{2^p}\right)$ pro $p \leq \log n - \log\log n$

(pro větší p platí taky, ale to je triviální)

porovnává dobu výpočtu s klasickým Bubblesortem, Hibbardovým Shellsortem a Quicksortem

třídí se náhodné posloupnosti reálných čísel generované z rovnoměrného rozdělení

délky posloupností $n = 10, 50, 100, 500, 1000, 2000, 10000$

čas se měří v milisekundách

programy byly napsány ve Fortranu a spuštěny na CDC Cyber 72

výsledky (jsou uvedeny v tabulce): nový algoritmus je výrazně rychlejší než klasický Bubblesort i Shellsort a jen o málo pomalejší než Quicksort

I když experimentální analýza nebyla provedena nijak závratným zpracováním (tím je představení nového algoritmu), přesto tomu něco chybí:

a) měří se pouze čas, nikoli teoretické charakteristiky třídění (počet porovnání, počet výměn apod.)
čas může být více či méně zkreslen konkrétním operačním systémem
vazba na teoretickou analýzu je problematická

b) chybí komentář k údajům v tabulce – jsou to průměrné hodnoty a z kolika měření?

c) chybí i ta nejelementárnější statistická analýza (minimum, maximum, medián, směrodatná odchylka, ...)

d) chybí grafické znázornění průběhu naměřených časů

e) třídily se velmi krátké posloupnosti (to mohlo být dáno možnostmi tehdejších počítačů)

f) chybí interpretace výsledků, např. že nový algoritmus je rychlejší než Quicksort pro krátké posloupnosti do 1000 prvků, od 2000 je rychlejší Quicksort

g) nejsou formulovány řádné hypotézy, např. že podle naměřených časů je nový algoritmus vždy přibližně 2x rychlejší než Shellsort – mohl by mít tedy stejnou asymptotickou složitost (liší se jen v multiplikativní konstantě)?

Přednosti této publikace: stručné představení nového algoritmu
inspirující námět pro další výzkum
aktuální příspěvek – mezi zasláním do časopisu a vydáním uplynuly 3 měsíce!

AN EFFICIENT VARIATION OF BUBBLE SORT

Wlodzimierz DOBOSIEWICZ
Institute of Informatics, Warsaw University, 00-901 Warszawa, Poland

Received 15 May 1980

Algorithms, sorting

Bubble sort is well known for being one of the worst sorting algorithms. One way to improve its efficiency was described by Batcher [1], however it is so complex and the amount of bookkeeping so huge that this algorithm is not commonly used in single-processor programming.

Bubble sort can be improved in yet another way, which is similar to Shell's version of the insertion sort.

We select a sequence $h_1, ..., h_t$, where $h_t > 1$ and $t = O(\log n)$, n being the number of sorted elements.

In pass i ($i \leqslant t$) the vector to be sorted is traversed from left to right and items distant h_i from each other are compared and exchanged if necessary. After t passes a regular bubble sort should be used to complete the sorting process. This gives the following algorithm:

```
for ℓ := 1 to t do
begin
  inc := hℓ;
  for i := 1 to n − inc do
   if A[i] > A[i + inc]
   then A[i] ↔ A[i + inc] fi
end;
ℓ := n − 1;
while ℓ > 0 do
  begin
   k := 0;
   for i := 1 to ℓ do
    if A[i] > A[i + 1]
    then {A[i] ↔ A[i + 1]; k := i};
   ℓ := k − 1
  end;
```

Obviously, this algorithm works, as the second part of it is just the well-known bubble sort. Its average efficiency is quite good, as illustrated by the Table 1.

The above results were obtained by sorting real numbers generated by a uniform distribution random numbers generator. The tested programs were written in Fortran and run on a CDC Cyber 72. The sequence $h_1 = \lfloor \frac{1}{2}n \rfloor$, $h_{i+1} = \lfloor \frac{3}{4}h_i \rfloor$ was used. The code of Quicksort was copied from [2], while Shellsort was coded after the description given in [3] with increments of form $2^k - 1$. While the version of Quicksort is approximately the best achievable, there is no certainty whether this is true in the case of Shellsort – this algorithm depends on the choice of increments (see [3] for details).

The exact time complexity of the presented algorithm has yet to be determined. The author hopes that this short presentation will encourage some one to work on this problem.

Table 1
Running times of various sorting algorithms (times in ms)

n	Quicksort	Shellsort	Bubblesort	Modified Bubblesort
10	0.86	0.99	0.6	0.58
50	5.84	8.62	14.6	4.62
100	13.38	21.28	57.3	11.87
500	87.81	153.70	1456.6	84.80
1000	199.8	367.0		194.9
2000	441.2	858.2		450.5
10000	2696.8			2743.8

2) Přibližně o 10 let později:

M.A. Weiss: Empirical study of the expected running time of Shellsort (Computer Journal 34 (1991), 88-91)

Je to experimentální studie známých algoritmů (Shellův, Hibbardův, Knuthův a Sedgewickův Shellsort) se známou složitostí v nejhorším případě, ale neznámou očekávanou složitostí (to je dodnes nevyřešený problém).

Dále obsahuje: orientační srovnání podle časů s Heapsortem a standardním a optimalizovaným quicksortem

srovnání s předchozí experimentální studií Shellsortu

Algoritmy nejsou detailně popsané, předpokládá se, že jsou známé. Jsou uvedeny dosažené teoretické výsledky o složitosti a výsledky předchozí exp. studie:

teoretický: Shell $\Theta(n^2)$ nejhorší případ, $\Theta(n^{3/2})$ očekávaný

Hibbard, Knuth $\Theta(n^{3/2})$ nejhorší případ

Sedgewick $\Theta(n^{4/3})$ nejhorší případ

experimentálně: Hibbard, Knuth mezi $\Theta(n^{1,25})$ a $\Theta(n^{1,28})$

nebo $\Theta(n \log^2 n)$ očekávaný případ

V experimentech se měřil čas (pouze pro orientaci) a počet výměn prvků (pro účely analýzy).

neměřil se počet porovnání, což je poměrně chyba (viz Nádvorníková 1997)

Třídily se náhodné permutace délky 100 až 12 000 000 prvků.

Byl použit UNIXový náhodný generátor RANDOM (RAND se neosvědčil).

Test probíhal na počítačích SUN 3 workstation s 8MB paměti a VAX 785 s 128 Mb.

Je uveden počet experimentů pro permutace různých délek (řádově desetitisíce pro permutace délky do 10 000, tisíce pro délky do 1 000 000, stovky pro delší).

K naměřeným průměrům jsou uvedeny směrodatné odchylky.

Srovnávají se: a) různé algoritmy na posloupnostech stejných délek

b) hlavně rychlost růstu počtu výměn v závislosti na zvětšující se velikosti souborů.

naměřené hodnoty se porovnávají se staršími odhady $\Theta(n^{1,26})$ a $\Theta(n \log^2 n)$

a prokládají se nově navrženými křivkami řádu

$\Theta(n^{5/4})$ pro Hibbardův a Knuthův Shellsort

$\Theta(n^{7/6})$ pro Sedgewickův

Výsledky porovnání různých aproximací jsou v tabulkách.

Chybí grafické srovnání a srovnání nějakou statistickou hodnotou, např. reziduálním součtem čtverců (chyb).

Na základě experimentů je vyslovena hypotéza: Pro přírůstkové posloupnosti délky $O(\log n)$ se složitost v nejhorším případě řádu $\Theta(n^k)$ transformuje na očekávanou složitost řádu $\Theta(n^{\frac{k+1}{2}})$.

J. Incerpi, R. Sedgewick: Practical variations of Shellsort (Information Processing Letters 26 (1987/88), 37–43

zobecnění Dobosiewiczova algoritmu (experimentální studie)

Weiss: Prezentace výsledků

Table 1. Running times (in seconds) of various sorting algorithms

	Shellsort					Quicksort	
N	Shell's	Hibbard's	Knuth's	Sedgewick's	Heapsort	Standard	Optimised*
100	0.0024	0.0022	0.0021	0.0017	0.0042	0.0028	0.0024
1000	0.0354	0.0344	0.0306	0.0293	0.0557	0.0315	0.0259
10000	0.5890	0.5563	0.5000	0.4300	0.7165	0.3677	0.3153
100000	9.230	8.408	8.041	5.730	8.859	4.230	3.588
1000000	141.6	130.3	135.4	71.2	104.7	47.1	41.3

* Recursive with median-of-three partitioning and a cutoff of 10.

Table 2 Hibbard

N	Observed exchanges	Our fit	$\Theta(N^{1.26})$ fit	$\Theta(N\log^2 N)$ fit
100	347.6	360.2	400.7	−245.9
500	2950.0	2944.1	3044.4	291.1
1000	7200.8	7139.0	7291.0	2514.6
5000	55831.4	55871.1	55397.2	42232.7
10000	134658.1	134966.1	132673.9	116237.5
50000	1038169.2	1038814.1	1008062.8	1022314.1
100000	2500788.7	2497784.2	2414267.4	2486839.0
500000	19120679.6	19113501.4	18343726.9	18290748.7
1000000	45848881.9	45873300.0	43932444.6	42248509.9
12000000	1058423789.1	1052351112.6	1005897526.8	787733188.2

Table 3

Permutation sizes	Total number of sorts	Machines used
$10i$, $10 \leqslant i \leqslant 99$	16000	16 SUN 3 workstations
$100i$, $10 \leqslant i \leqslant 99$	16000	16 SUN 3 workstations
$1000i$, $10 \leqslant i \leqslant 45$	4000	16 SUN 3 workstations
$100000i$, $5 \leqslant i \leqslant 10$	2000	5 SUN 3 workstations with ≥ 8 Mb
12000000	100	1 VAX 785 with 128 Mb

Table 4

	Hibbard's		Knuth's		Sedgewick's	
N	Average	S.D.	Average	S.D.	Average	S.D.
100	347.6	26.3	432.1	34.3	461.8	37.9
1000	7200.8	361.8	8901.3	446.8	7725.2	192.0
10000	134658.1	6485.6	164960.3	7218.7	108811.7	943.0
100000	2500788.7	116348.3	2963566.6	1331956.0	1409404.5	5281.4
1000000	45848881.9	2.045907.5	53477706.7	2306321.8	17421118.3	36068.8

Table 6

Exchanges

N	Hibbard's Observed	Hibbard's Our fit	Knuth's Observed	Knuth's Our fit	Sedgewick's Observed	Sedgewick's Our fit
100	347.6	360.2	432.1	381.7	461.8	456.3
200	879.5	915.9	1 094.9	1 088.3	1 110.4	1 092.4
300	1 499.3	1 536.8	1 863.2	1 873.5	1 817.1	1 810.6
400	2 184.5	2 215.8	2 725.1	2 727.8	2 598.2	2 577.8
500	2 950.0	2 944.1	3 608.3	3 640.6	3 410.9	3 380.3
600	3 692.7	3 715.1	4 583.7	4 603.9	4 210.7	4 210.3
700	4 516.7	4 523.6	5 606.5	5 611.6	5 054.9	5 063.1
800	5 370.8	5 365.8	6 640.8	6 659.1	5 933.0	5 935.4
900	6 264.6	6 238.4	7 729.2	7 742.6	6 847.4	6 824.5
1 000	7 200.8	7 139.0	8 901.3	8 859.3	7 725.2	7 772.8
2 000	17 480.6	17 338.6	21 395.7	21 433.8	17 401.2	17 360.2
3 000	29 171.5	29 166.6	36 035.1	35 865.6	27 632.1	27 700.0
4 000	42 300.4	42 036.6	51 670.2	51 636.0	38 572.8	38 503.0
5 000	55 831.4	55 871.1	68 128.9	68 475.1	49 447.3	49 652.4
6 000	70 411.6	70 477.2	85 966.8	86 213.6	61 008.9	61 079.1
7 000	85 667.8	85 755.5	104 575.3	104 734.0	72 748.0	72 737.1
8 000	102 007.0	101 632.5	123 897.5	123 949.5	84 929.4	84 593.5
9 000	117 931.4	118 051.2	143 972.7	143 793.2	96 961.9	96 624.0
10 000	134 658.1	134 966.1	164 960.3	164 211.2	108 811.8	108 809.2
20 000	324 773.0	325 331.3	392 459.0	392 865.7	236 327.7	236 779.2
30 000	543 129.3	543 925.2	654 918.0	653 937.4	327 777.2	372 092.5
40 000	781 219.3	783 095.2	934 984.3	938 534.9	512.798.8	512 196.8
50 000	1 038 169.2	1 038 814.1	1 239 432.9	1 242 001.8	655 480.2	655 873.8
60 000	1 309 707.4	1 308 534.4	1 562 018.2	1 561 402.6	803 536.0	802 404.1
70 000	1 589 450.2	1 590 476.7	1 895 389.6	1 894 691.7	951 420.6	951 311.4
80 000	1 882 107.7	1 883 317.7	2 240 797.3	2 240 350.7	1 100 403.4	1 102 256.9
90 000	2 183 821.4	2 186 028.8	2 606 280.3	2 597 201.2	1 252 909.2	1 254 986.8
100 000	2 500 788.7	2 497 784.2	2 963 566.6	2 964 298.7	1 409 404.5	1 409 304.2
200 000	6 000 092.0	6 002 478.5	7 092 600.7	7 072 978.0	3 013 151.4	3 015 182.2
300 000	10 023 436.6	10 022 280.4	11 810 720.0	11 762 684.6	4 690 850.4	4 696 427.9
400 000	14 429 076.3	14 417 206.3	16 910 384.1	16 874 618.7	6 425 108.0	6 427 022.2
500 000	19 120 679.6	19 113 501.4	22 408 676.5	22 325 341.3	8 209 540.0	8 194 562.7
600 000	24 056 494.9	24 064 514.9	28 156 209.6	28 062 164.4	10 015 607.2	9 991 722.2
700 000	29 236 932.2	29 237 678.8	34 206 881.1	34 048 349.0	11 811 384.4	11 813 644.9
800 000	34 603 915.5	34 608 781.6	40 458 926.4	40 256 606.9	13 649 817.6	13 656 870.6
900 000	40 198 106.3	40 159 017.8	46 810 624.1	46 665 758.2	15 521 352.5	15 518 805.5
1 000 000	45 848 881.9	45 873 300.0	53 477 706.7	52 258 820.7	17 421 118.3	17 397 431.6
12 000 000	1 058 423 789.1	1 052 351 112.6	1 209 048 063.7	1 201 162 111.6	254 835 978.7	254 636 241.3

by using a poor random number generator or time lost due to crashes. By running these sorts simultaneously, only about four weeks of actual calendar time was used.

3. Increments which give $O(N^{\frac{3}{2}})$ worst-case bounds

In this section we provide new fits for the running time of Shellsort using the increment sequences suggested by Hibbard and Knuth. The main measure of the running time is the number of exchanges performed by the algorithm. In both cases, the number of exchanges fits $c_1 N^{\frac{5}{4}} + c_2 N + c_3 N^{\frac{3}{4}} + c_4 N^{\frac{1}{2}} + c_5 N^{\frac{1}{4}} + c_6$. In addition to this, there is additional $O(N \log N)$ work corresponding to comparisons which do not require exchanges (one comparison per element per increment). This gives a total running tome of $O(N^{\frac{5}{4}})$. Table 6 shows how well our fit compares with the observation.

The number of exchanges observed has a high standard deviation for both of the sequences in this section; in particular, for the sequence recommended by Knuth, the standard deviation is about 5% of the observed value (the claim in Ref. 8 of a standard deviation of 50 000 for sorting 250 000 numbers is apparently a typo that should read 350 000).

All of the formulas represent simple unweighted least-mean-square fits, and it is certainly possible that some other technique could produce even better fits. The fact that a simple method produces such good fits over such a wide range shows how likely the functional form is. It seems likely that the error in the fit was larger than the probable error in the observation. The actual constants are of little practical interest, probably insignificant and inaccurate, but for the sake of completeness we include them here. For Hibbard's sequence, the fit obtained is

$$1.55376872N^{\frac{5}{4}} - 4.47754N + 47.98950721N^{\frac{3}{4}} - 335.828N^{\frac{1}{2}} + 1140.81404N^{\frac{1}{4}} - 1450.25$$

This is in contrast to the fits $1.21N^{1.26}$ and $0.39N \log N - 2.33N \log N$ previously proposed. For Knuth's sequence, the fit is

$$1.73123269N^{\frac{5}{4}} - 2.28388N + 33.14353571N^{\frac{3}{4}} - 284.216N^{\frac{1}{2}} + 1074.63905N^{\frac{1}{4}} - 1541.59,$$

which contrasts with Knuth's conjecture of $1.66N^{1.25}$ and a logarithmic form that is not plausible.

4. Sedgewick's increments ($h_i = 9.4^i - 9.2^i + 1 \cup 4^i - 3.2^i + 1$)

These increments exhibit a very low standard deviation which enables us to get fits with much less error than in the previous section. Table 4 compares the average number and standard deviation (S.D.) of exchanges for Hibbard's, Knuth's and Sedgewick's increment sequences on a few file sizes.

Our fit for this sequence is $0.42663452N^{\frac{7}{6}} + 18.49281148N - 61.5728N^{\frac{5}{6}} + 72.69723933N^{\frac{2}{3}} + 105.37474557N^{\frac{1}{2}} - 372.755N^{\frac{1}{3}} + 483.29055$ *TLS*10 115. The $O(N^{\frac{1}{6}})$ term was too insignificant to be reported. The data in tables 5 and 6 shows that this is a very accurate fit. Attempts to fit using other functional forms (including the form in Section 3) give generally poor results.

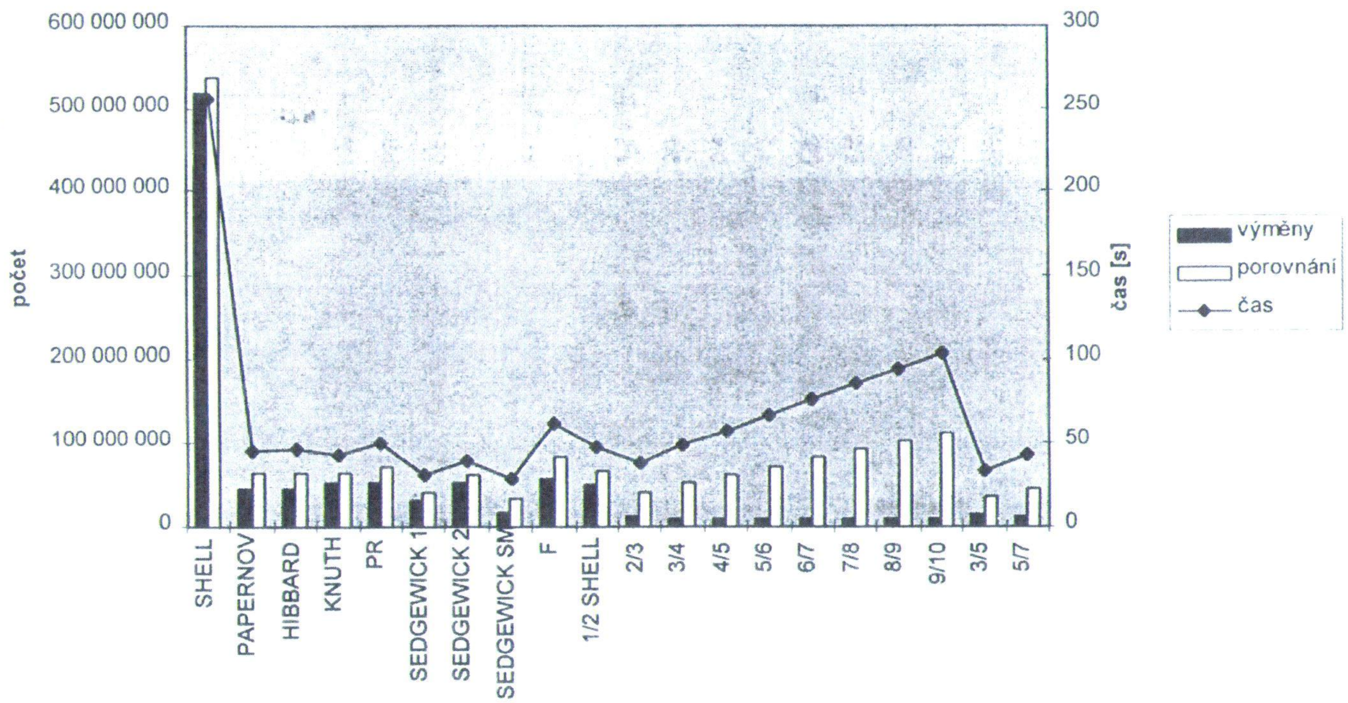
5. Extensions, conjectures and open problems

The natural open problem is to prove any of

T. Nádvorníková - dipl práce (1997)

Graf 19 - Porovnání měření pro různé verze - permutace délky 1 000 000

Graf zobrazuje průměr naměřených hodnot. V grafu vidíme poměr počtu výměn prvků vůči počtu porovnání prvků a času. U verze Sedgewick 2 bylo vysoké procento prvků po porovnání vyměněno, naopak verze 8/9 a 9/10 prováděly zbytečně moc nevyužitých porovnání (tj. porovnání, po kterých nenásledovala výměna). Naměřený čas koresponduje s počtem porovnání.



3) Po dalších zhruba 10 letech:

S. Hanke: The performance of concurrent red-black tree algorithms

(TR 115, Institut für Informatik, Univ. Freiburg, 1998, 45 stran)

Rozsáhlá experimentální studie zaměřená na vzájemné porovnání 3 typů tzv. relaxovaných červeno-černých stromů a klasického červeno-černého stromu.

Relaxovaný vyvážený červeno-černý strom je binární vyhledávací strom s poněkud slabší podmínkou na vyváženost, která sice o něco zhorší parametry stromu (a tím i rychlost vyhledávání, vkládání atd.), ale kompenzuje to menším počtem vyvažovacích operací, které navíc probíhají jen v bezprostředním okolí nějakého vrcholu a neovlivňují zbytek. To je výhoda při vysoké zátěži a paralelním přístupu uživatelů - nemusí se čekat, až se po každé jednotlivé operaci strom klasicky vyváží, což by znamenalo zablokovat jeho velkou část. Struktura navíc uchovává další parametry, pomocí nichž je možné v době slabého provozu strom dovyvážit na klasický červeno-černý strom.

Studie obsahuje: úvod do problematiky a rešerši dosavadních výsledků

definice daných struktur (standardní červeno-černý strom + algoritmy pro 3 typy relaxovaného vyvažování)

implementační detaily (způsob obsluhy vyvažovacích pořadačů, paralelní přístup - zamykání)

popis simulace paralelního prostředí (1, 3, 7, 15, 31 a 63 uživatelských procesů + 1 proces na pořadač pro vyvažovací operace) CRCW-MIMD se sdílenou pamětí a s jednotkovou cenou operací,

synchronisace pomocí semaforů a zamykání

V experimentech bylo provedeno vždy 1000 000 slovníkových operací (search, insert, delete) paralelně na původně vytvořeném červeno-černém stromě s 1000 000 klíči (vytvořeném postupným vkládáním náhodně vybraných klíčů do původně prázdného stromu).

pravděpodobnosti operací: 0,5 pro insert
0,3 pro search
0,2 pro delete

Experimenty se opakovaly s různými počty procesů.

Byla volena cena zamykacích operací 1, 10, 20, 30, 40, 50, 60 jednotek času.

Dále následuje obsáhlý popis různých typů experimentů (co se měřilo, v závislosti na čem, s čím se porovnávalo) se spoustou dvou- a třírozměrných grafů. Nějaké tabulky naměřených hodnot už se neuvádějí.

Komentáři jenom stručně popisuje to, co je na obrázcích, úvahy typu Proč to tak dopadlo, dalo se to čekat nebo je to překvapující? o podstatě chyb, statistické zpracování výsledků chybí.

Závěr je velmi stručný ve smyslu, že relaxované vyvažování v konkurenčním prostředí se osvědčilo.

Hanke: ukázka prezentace výsledků

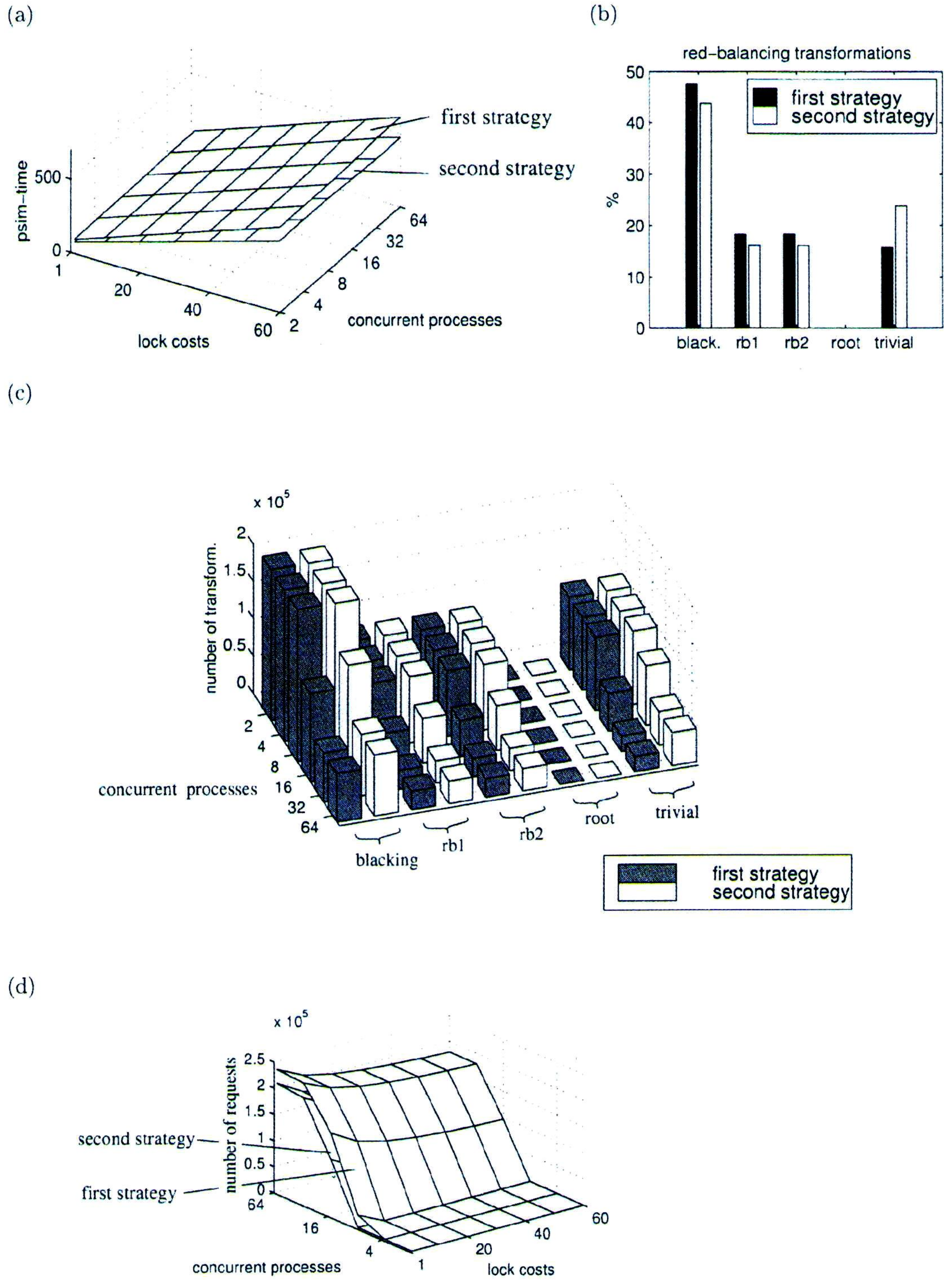


Figure 28: Comparison of the two strategies to handle red-red conflicts, exemplary shown by the chromatic tree: (a) Average time needed to perform a red-balancing transformation. (b) Proportion of the performed red-balancing transformations (using 63 user processes and 1 rebalancing process). (c) Total number of performed red-balancing transformations. (d) Number of unsettled rebalancing requests after the time used to perform 1000000 dictionary operations.

4) V novém tisíciletí:

Ch. Barrett et al.: Statistical analysis of algorithms: A case study of market-clearing mechanisms in the power industry

(Journal of Graph Algorithms and Applications 7 (2003), 3-31

Jedná se (kalím) málo studií, kde se výsledky z experimentů statisticky vyhodnocovaly.

Řeší se praktický problém související s deregulací elektrického průmyslu v USA, řešení vztahů mezi producenty a odběrateli v podmínkách, kdy mnozí sdílejí jedinou rozvodnou síť s omezenou kapacitou.

Protože je to komplikované, byla ve většině států vytvořena instituce nezávislého systémového operátora či arbitra, který má rozhodovat, které kontrakty se povolí a které ne - za tím účelem byla stanovena nějaká kritéria. Teoreticky se pak řeší ještě grafový problém (toky v sítích s omezujícími podmínkami?).

Testovaly se 4 algoritmy, z toho 3 heuristiky.

Jako síť se zvolila energetická síť státu Colorado:

zdroje - elektrárny, jejich umístění a kapacity

spotřebiče - odhadly se na základě demografie

kapacity hran - odhadly se z kapacit zdrojů následovně:

celková kapacita spotřebičů se rovná celkové kapacitě zdrojů a je rozdělena mezi spotřebiče podle počtu obyvatelstva

kapacity hran se určí simulací - síť se prožene algoritmy na výpočet max. toku kontakty se generují podle různých scénářů – jsou popsány 4

V experimentech se pro každý scénář zkoušely všechny 4 algoritmy.

Byl použit software a formát dat projektu DIMACS.

Měřila se doba výpočtu a kvalita řešení, přičemž kvalita byla rozhodující a byla pro ni stanovena jakási vyjádřitelná míra.

K testování rozdílů mezi jednotlivými scénáři a algoritmy byla použita analýza rozptylu.

V práci je poměrně důkladně popsaná metoda analýzy rozptylu – na rozdíl od jiných studií, kde se prezentují pouze výsledky vypadlé na zmáčknutí knoflíku z nějakého statistického softwaru jako z černé skříňky. Uživatel tak má šanci zjistit, jestli ta použitá metoda je vůbec pro jeho data vhodná!

Na ukázku uvádím pouze část netriviální statistické analýzy scenarios perform the best.[1]

ANOVA has the following three advantages over individual t-tests[2] when the number of groups being compared is greater than two. See [9] for more details. In our case, we have four algorithms and four scenarios. Standard statistics terminology for a hypothesis that we wish to test, is *null hypothesis*.

- It gives accurate and known type-I error probability.[3]
- It is more powerful i.e. if null hypothesis is false, it is more likely to be rejected.
- It can assess the effects of two or more independent variables simultaneously.

5.3 Mathematical Model

Quality of Solution: We first describe the experiment for the quality of solution i.e. $p_{\mathcal{AS}}$. We use a two-factor ANOVA model since our experiment involves two factors which are:

1. The algorithms: $\mathcal{A}_i, i = 1, 2, 3$ and 4.
2. The scenario: $\mathcal{S}_j, j = 1, 2, 3$ and 4.

Following classical statistics terminology, we will sometimes refer to algorithms as *treatments* and the scenarios as *blocks*. We will use $\mathcal{A}$ to denote the set of algorithms and $\mathcal{S}$ to denote the set of scenarios. For each algorithm-scenario pair we have *30 observations (or replicates)*. When testing the efficacy of the algorithms, we use 4 algorithms, each having 120 observations (30 for each scenario) from the corresponding population. The design of experiment used here is a *fixed-effect complete randomized block*. *Fixed-effect* because the factors are fixed as opposed to randomly drawn from a class of algorithms or scenarios; the conclusions drawn from this model will hold only for these particular algorithms and scenarios. *Complete* implies that the number of observations are the same for each block. *Randomized* refers to the 30 replicates being drawn randomly. We wish to test the hypothesis:

---

[1] The populations in each of the groups are assumed to be normally distributed and have equal variances. The effect of violation of ANOVA assumptions of normality and homogeneity of variances have been tested in the literature ([10]) and the results show:

- Non-normality has negligible consequences on type-I and II error probabilities unless the populations are highly skewed or the sample is very small.
- When the design is balanced, i.e. the number of observations are the same for each group, violation of homogeneity of variance assumption has negligible consequences on the accuracy of type-I error probabilities.

[2] t-test checks for the significance of the difference in the means of *two* samples. It can assess whether the difference in sample means is just due to sampling error or they really are from two populations with different means.

[3] The probability of rejecting a null hypothesis when it is actually true.

> Is the mean quality of solution provided by different algorithms the same, against the alternative hypothesis that some or all of these means are unequal?

The model for randomized block design includes constants for measuring the scenario effect (block effect), the algorithm effect (treatment effect) and a possible interaction between the scenarios and the algorithms. The appropriate mathematical model is as follows:

$$X_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \varepsilon_{ijk},$$

where X_{ijk} is the measurement ($p_{\mathcal{AS}}$) for the kth sample within the ith algorithm and the jth scenario. τ_i is the algorithm effect. β_j is the scenario effect. $(\tau\beta)_{ij}$ captures the interaction present between the algorithms and the scenarios. ε_{ijk} is the random error. See [8, 9] for further details on ANOVA.

We use S-Plus [15] software to run two-factor ANOVA to test the following three different null hypotheses.

1. Are the means given by the 4 different algorithms equal? The null hypothesis here is, $H_0 : \tau_1 = \tau_2 = \tau_3 = \tau_4$.
2. Are the means given by the 4 different scenarios equal? The null hypothesis here is, $H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4$.
3. Is there any interaction between the two factors? The null hypothesis here is, $H_0 : (\tau\beta)_{ij} = 0$.

The results of two-factor ANOVA are shown in Table 1 and Table 2. In the following discussion, we explain the meaning of each column in Table 1. DF refers to the degrees of freedom, SS refers to the sum of squared deviations from the mean. MS refers to the mean square error, which is the sum of squares divided by the degrees of freedom.[4]

[4] The sum of squares for the algorithm factor can be calculated as:

$$SS_{\mathcal{A}} = nJ\Sigma_i(\overline{X}_{i..} - \overline{X}_{...})^2$$

where n is the number of replicates, J is the number of scenarios, $\overline{X}_{i..}$ is the mean of algorithm i across all scenarios and $\overline{X}_{...}$ is the grand mean across all algorithms and scenarios. Recall that in our case $n = 30$ and $J = 4$ yielding a total sample size of 120.

The sum of squares for scenario factor can be calculated as:

$$SS_{\mathcal{S}} = nI\Sigma_j(\overline{X}_{.j.} - \overline{X}_{...})^2$$

where as before n is the number of replicates, I is the number of algorithms and $\overline{X}_{.j.}$ is the mean of scenario j across all algorithms. Again, in our case $n = 30$ and $I = 4$.

The sum of squares for algorithms and scenario interaction is:

$$SS_{\mathcal{AS}} = n\Sigma j\Sigma_i[\overline{X}_{ij.} - (\overline{X}_{...} + \hat{\tau}_i + \hat{\beta}_j)]^2$$

Here $\overline{X}_{ij.}$ is the mean of observations for the algorithm i scenario j pair. $\hat{\tau}_i$ and $\hat{\beta}_j$ are respectively the estimated least square values of τ_i and β_j. The sum of squares "within" refers to the squared difference between each observation and the mean of the scenario and algorithm of which it is a member. It is also referred as the residual sum of squares. This can be calculated as:

$$SS_{\mathcal{W}} = n\Sigma j\Sigma_i\Sigma_k(X_{ijk} - \overline{X}_{ij.})^2$$

The p-value gives the smallest level of significance at which the null hypothesis can be rejected.[5] The lower the p-value, the lesser the agreement between the data and the null hypothesis. Finally the F-**test** is as follows. To test the null hypothesis, i.e., whether the population means are equal, ANOVA compares two estimates of σ^2. The first estimate is based on the variability of each population mean around the grand mean. The second is based on the variability of the observations in each population around the mean of that population. If the null hypothesis is true, the two estimates of σ^2 should be essentially the same. Otherwise, if the populations have different means, the variability of the population mean around the grand mean will be much higher than the variability within the population. The null hypothesis in the F-test will be accepted if the two estimates of σ^2 are almost equal.

In a two-factor fixed-effect ANOVA, three separate F-**tests** are performed: two tests for the factors, and the third for the interaction term. The null hypothesis for the first factor can be written as:

$$H_0^{\mathcal{A}} : \mu_{1\cdot\cdot} = \mu_{2\cdot\cdot} = \cdots = \mu_{j\cdot\cdot}$$

which is equivalent to writing: $H_0 : \tau_1 = \tau_2 = \tau_3 = \tau_4$. The F-test is:

$$F_{\mathcal{A}} = \frac{SS_{\mathcal{A}}/(I-1)}{SS_{\mathcal{W}}/IJ(n-1)}$$

and the null hypothesis for the second factor can be written as:

$$H_0^{\mathcal{S}} : \mu_{\cdot 1\cdot} = \mu_{\cdot 2\cdot} = \cdots = \mu_{\cdot j\cdot}$$

which is equivalent to writing: $H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4$. The F-test is:

$$F_{\mathcal{S}} = \frac{SS_{\mathcal{S}}/(J-1)}{SS_{\mathcal{W}}/IJ(n-1)}$$

and the null hypothesis for the interaction term can be written as:

$$H_0^{\mathcal{AS}} : (\tau\beta)_{ij} = 0.$$

The F-test is:

$$F_{\mathcal{AS}} = \frac{SS_{\mathcal{AS}}/(I-1)(J-1)}{SS_{\mathcal{W}}/IJ(n-1)}$$

If this F-ratio is close to 1, the null hypothesis is true. If it is considerably larger – implying that the variance between means is larger than the variance

The total sum of squares is

$$SS_{\mathcal{T}} = SS_{\mathcal{A}} + SS_{\mathcal{S}} + SS_{\mathcal{AS}} + SS_{\mathcal{W}}$$

[5]To obtain a p-value for say $F_{\mathcal{A}}$, the algorithm effect, we would look across the row associated with 3 degree of freedom in the numerator and 464 degrees of freedom in the denominator in the F-distribution table and find the largest value that is still less than the one obtained experimentally. From this value, we obtain a p-value of 0 for $F_{\mathcal{A}}$.

Source	DF	SS	MS	F-test	p-value
Scenario (Block)	3	0.14	0.05	43.38	0
Algorithm (Treatment)	3	22.78	7.59	6792.60	0
Scenario:Algorithm	9	0.12	0.01	15.90	0
Residuals	464	0.40	.0008		
Total	479	23.45			

Table 1: **Results of Two-Factor ANOVA:** This table shows results of two-factor ANOVA where the factors are algorithms and scenarios. The measurement is the quality of solution, given by $p_{\mathcal{AS}}$. The p-values show that the algorithm effect, scenario effect and the interaction between the algorithms and scenarios are all significant at any level of confidence.

within a population – the null hypothesis is rejected. The F distribution table should be checked to see if the F-ratio is significantly large.

The results in Table 1 show that all the above three null hypothesis are rejected at any significance level. This implies that the performance (measured by $p_{\mathcal{AS}}$) of at least one of the algorithms is significantly different from the other algorithms. Also, different scenarios make a difference in the performance. Finally, the scenarios and the algorithms interact in a significant way. The interaction implies that the performance of the algorithms are different for different scenarios.

5.3.1 Contrasts

The next question of interest is what really caused the rejection of the null hypothesis; just knowing that at least one of the algorithms is different does not help us identify which algorithm is significantly different. To answer this we use a procedure called *contrast.* A contrast $\mathcal{C}$ among I population means (μ_i) is a linear combination of the form

$$\mathcal{C} = \Sigma_i \alpha_i \mu_i = \alpha_1 \mu_1 + \alpha_2 \mu_2 + \cdots + \alpha_I \mu_I$$

such that the sum of contrast coefficients $\Sigma_i \alpha_i$ is zero. In the absence of true population means, we use the unbiased sample means which gives the estimated contrast as:

$$\hat{\mathcal{C}} = \Sigma_i \alpha_i \overline{X}_i = \alpha_1 \overline{X}_1 + \alpha_2 \overline{X}_2 + \cdots + \alpha_I \overline{X}_I.$$

The contrast coefficients α_1, α_2, $\cdots$, α_I are just positive and negative numbers that define the particular hypothesis to be tested. The null hypothesis states that the value of a parameter of interest for every contrast is zero, i.e., $H_0 : \mathcal{C} =$