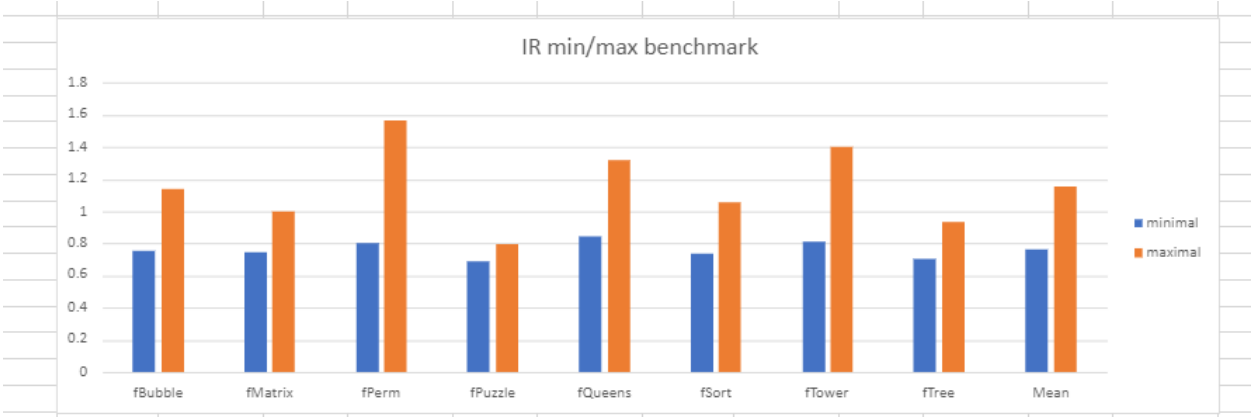
Lab 2 – OLDSIM

Benchmark	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fSort	fTower	fTree	Mean
minimal	0.75749276	0.747852	0.80177966	0.688183	0.84256156	0.7358521	0.81353028	0.701682	0.76111673
maximal	1.14058348	1.004132	1.56557817	0.798885	1.31866587	1.0571529	1.39933028	0.933834	1.15227021

File	FBubble		fMatrix		fPerm		fPuzzle		fQueens		fSort		fTower		fTree	
Benchmark	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions	Cycles	Instructions
minimal	271996	206035	309973	231814	443567	355643	10000001	6881833	244991	206420	97983	72101	308715	251149	193877	136040
maximal	180640	206035	230860	231814	227164	355643	8614302	6881833	156537	206420	68203	72101	179478	251149	145679	136040



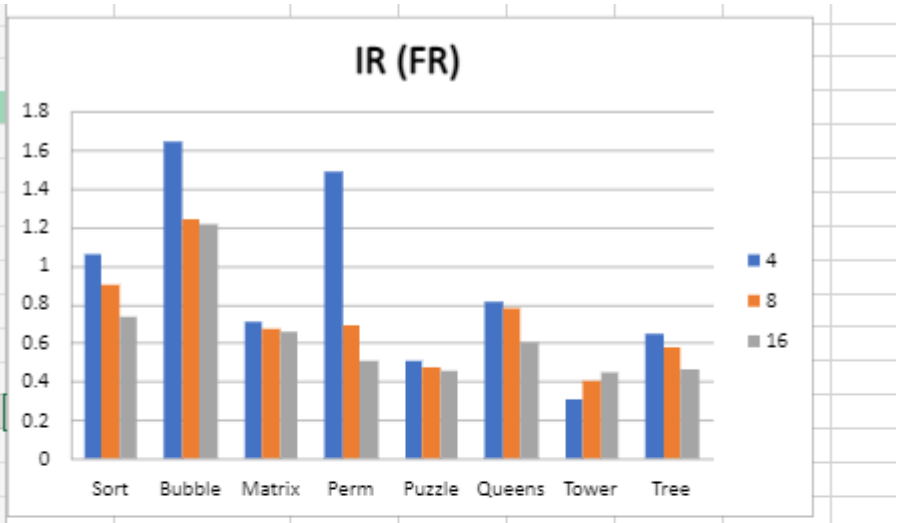
Se observa ca IR(Instruction Rate) pentru modelul maximal este mai mare decat cel pentru modelul minimal, pentru fiecare din cele 8 benchmark-uri. Desi acesta nu este cu mult mai mare, a durat mult mai mult timp executia acestuia.

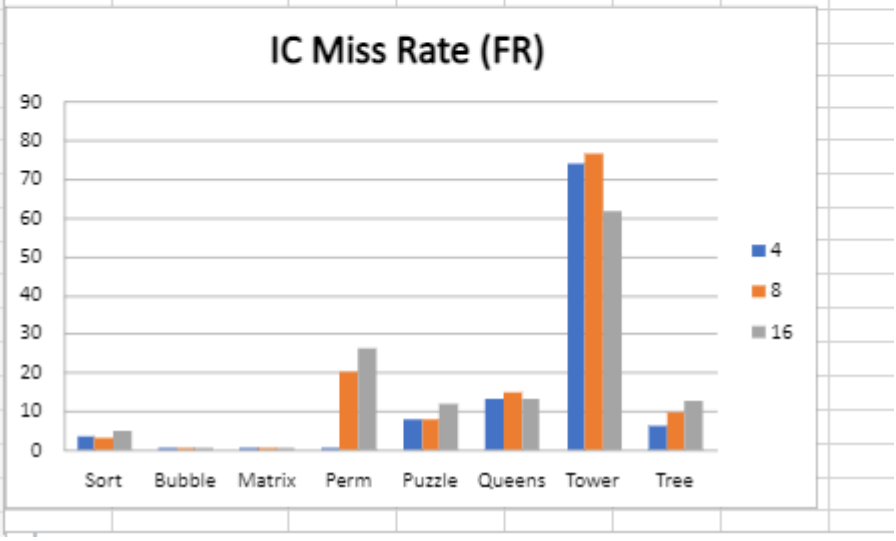
Lab 3 - SIMCACHE

Ex. 1

IR (Issue Rate)								
FR	Sort	Bubble	Matrix	Perm	Puzzle	Queens	Tower	Tree
4	1.058	1.648	0.705	1.483	0.505	0.818	0.308	0.651
8	0.904	1.242	0.678	0.692	0.474	0.777	0.401	0.578
16	0.737	1.218	0.654	0.505	0.455	0.608	0.45	0.462

IC Miss Rate (%)								
FR	Sort	Bubble	Matrix	Perm	Puzzle	Queens	Tower	Tree
4	3.5	0.05	0.05	0.03	7.79	13.3	74.04	6.06
8	3.09	0.04	0.05	20.38	7.94	14.82	76.46	9.52
16	4.7	0.05	0.06	26.13	12	13.2	61.7	12.94



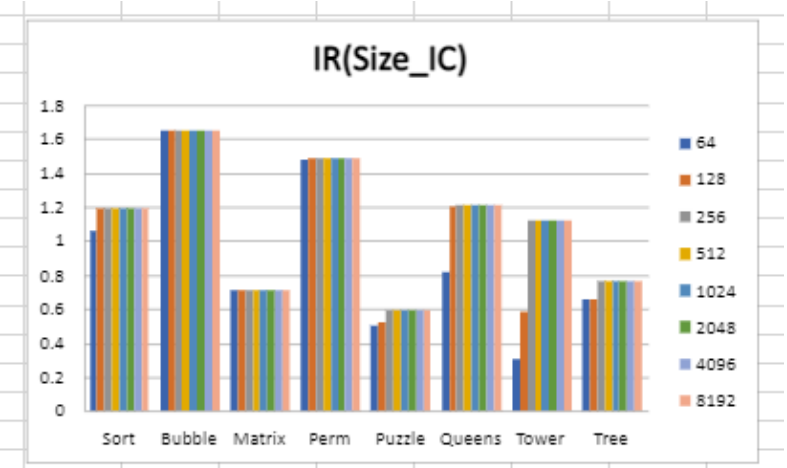


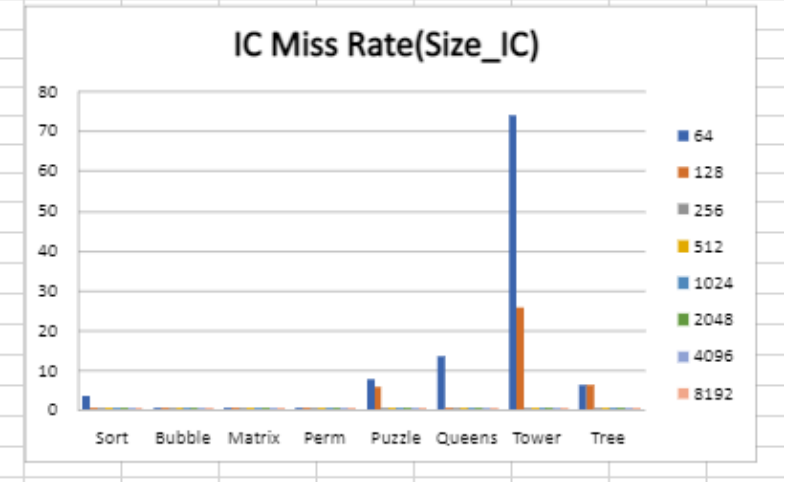
Se observa ca odata cu cresterea Fetch Rate-ului, rata de procesare a benchmark-urilor scade (mai putin Tower, care creste) iar IC Miss Rate creste sau ramane tot pe acolo, mai putin Tower care scade

EX. 2

IR (Issue Rate)								
Size_IC	Sort	Bubble	Matrix	Perm	Puzzle	Queens	Tower	Tree
64	1.058	1.648	0.705	1.483	0.505	0.818	0.308	0.651
128	1.196	1.648	0.706	1.484	0.524	1.201	0.586	0.651
256	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759
512	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759
1024	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759
2048	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759
4096	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759
8192	1.196	1.648	0.706	1.484	0.595	1.207	1.12	0.759

IC Miss Rate (%)								
Size_IC	Sort	Bubble	Matrix	Perm	Puzzle	Queens	Tower	Tree
64	3.5	0.05	0.05	0.03	7.79	13.3	74.04	6.06
128	0.18	0.05	0.05	0.02	6	0.2	25.65	6.06
256	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1
512	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1
1024	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1
2048	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1
4096	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1
8192	0.17	0.05	0.05	0.02	0.05	0.07	0.08	0.1

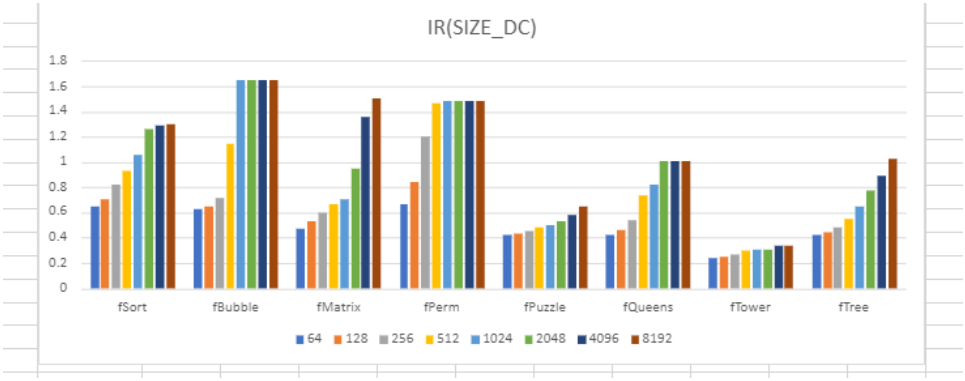




Se observa ca atat IR creste odata cu dimensiunea cache-ului de instructiuni, iar rata de miss in cache-ul de instructiuni scade. De asemenea, de la dimensiunea de 256 valorile raman aceleasi pana la 8192, prin urmare nu se merita sa implementam un cache de instructiuni mai mare de 256 de locatii.

Ex. 3

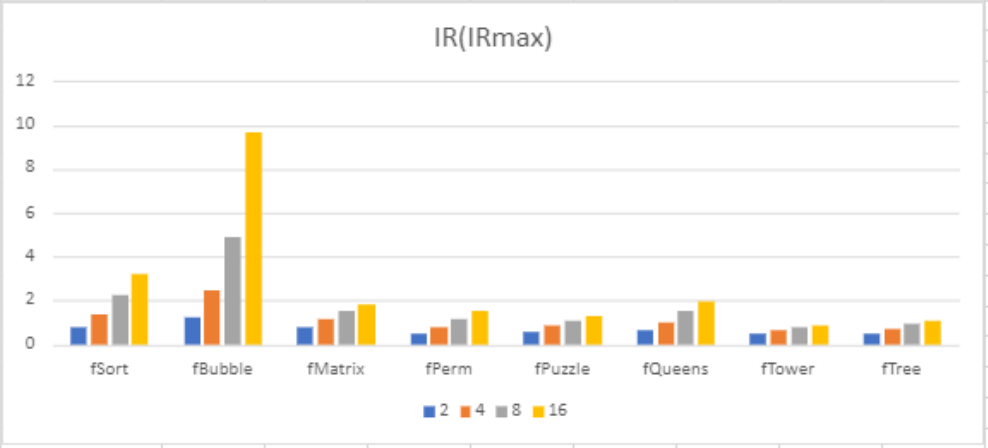
IR(SIZE_DC)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
64	0.648	0.632	0.468	0.664	0.428	0.428	0.241	0.419
128	0.707	0.647	0.532	0.845	0.433	0.462	0.252	0.44
256	0.82	0.715	0.595	1.206	0.454	0.543	0.264	0.48
512	0.928	1.139	0.67	1.468	0.481	0.733	0.297	0.546
1024	1.058	1.648	0.705	1.483	0.505	0.818	0.308	0.651
2048	1.257	1.649	0.948	1.483	0.529	1.009	0.308	0.77
4096	1.288	1.649	1.354	1.483	0.576	1.009	0.334	0.888
8192	1.297	1.649	1.5	1.483	0.649	1.009	0.334	1.027



Se observa o crestere constanta a IR pana la 2048 de locatii in cache-ul de date. Totusi, IR-ul unor benchmark-uri precum Tree, Matrix si Puzzle prezinta o crestere si la valori mai mari de 2048.

Ex. 4

IR(IRmax)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
2	0.777	1.219	0.808	0.505	0.564	0.635	0.471	0.468
4	1.373	2.435	1.178	0.811	0.826	1.034	0.639	0.69
8	2.227	4.861	1.528	1.163	1.075	1.507	0.778	0.903
16	3.223	9.685	1.795	1.484	1.265	1.954	0.872	1.068

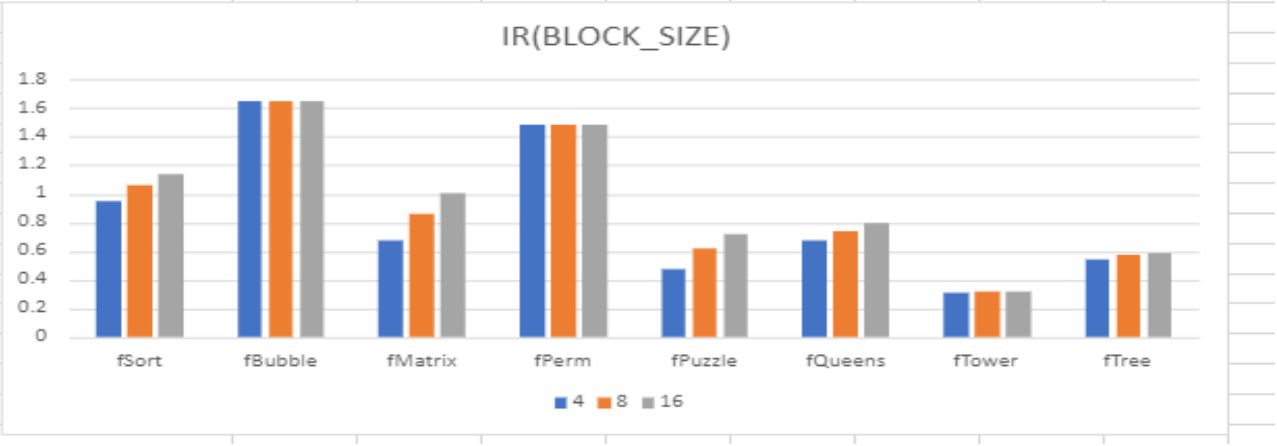


Majoritatea benchmark-urilor nu prezinta o crestere semnificativa a IR in functie de IRMax, exceptie facand Bubble si Sort.

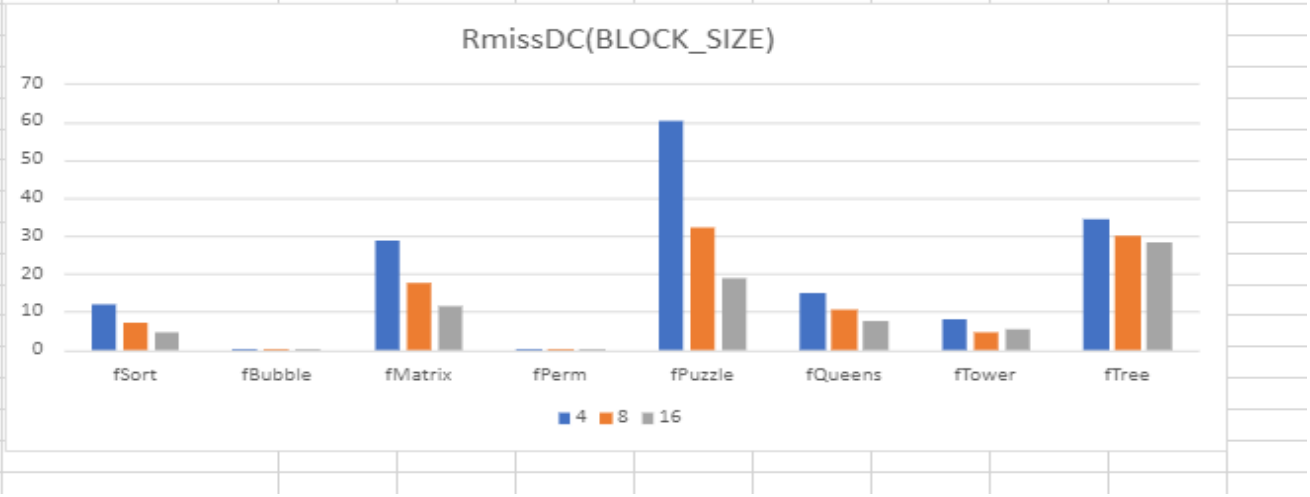
Ex. 5

Write Back

IR(BLOCK_SIZE)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
4	0.949	1.646	0.669	1.483	0.478	0.674	0.303	0.534
8	1.062	1.647	0.858	1.483	0.619	0.742	0.315	0.569
16	1.143	1.647	1.009	1.483	0.72	0.8	0.312	0.583

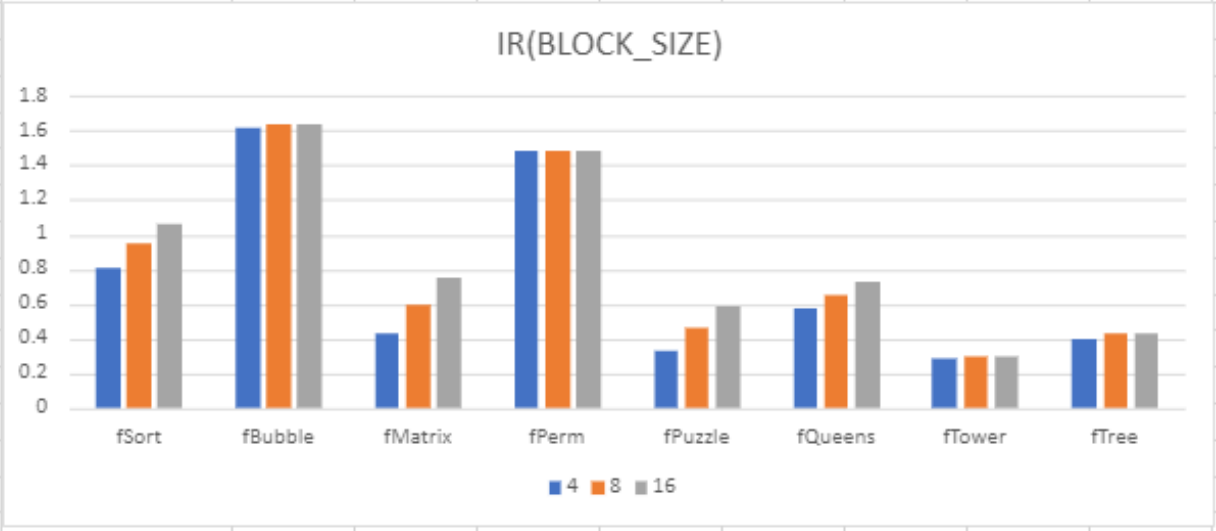


RmissDC(BLOCK_SIZE)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
4	12.1	0.36	28.84	0.04	60.21	14.75	7.93	34.54
8	7.29	0.19	17.39	0.02	32.16	10.68	4.63	30.14
16	4.46	0.1	11.31	0.01	18.81	7.71	5.55	28.36

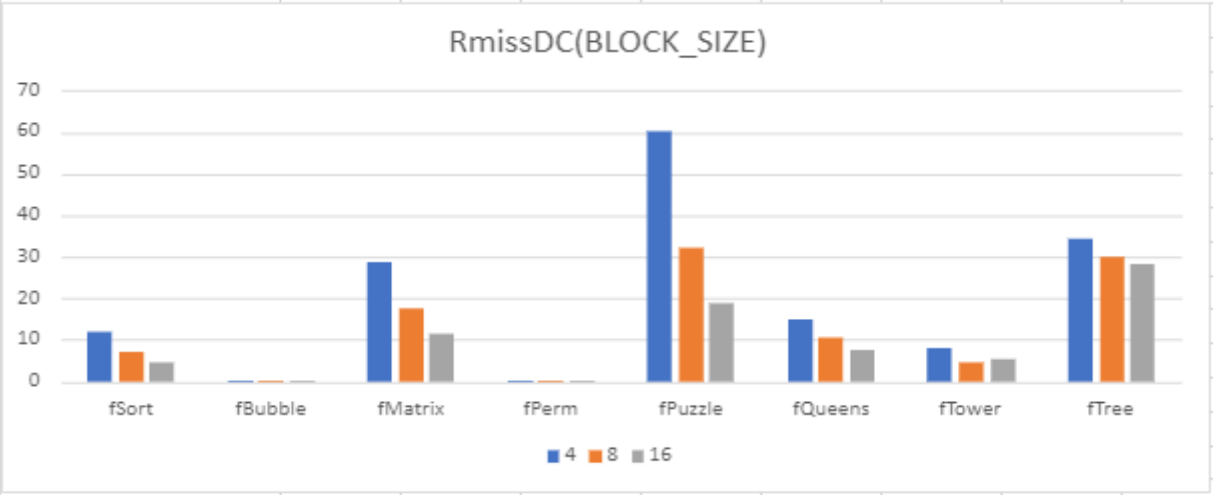


Write Through

IR(BLOCK_SIZE)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
4	0.801	1.619	0.43	1.48	0.323	0.574	0.279	0.398
8	0.945	1.633	0.598	1.481	0.466	0.655	0.299	0.423
16	1.056	1.64	0.753	1.482	0.589	0.726	0.294	0.432



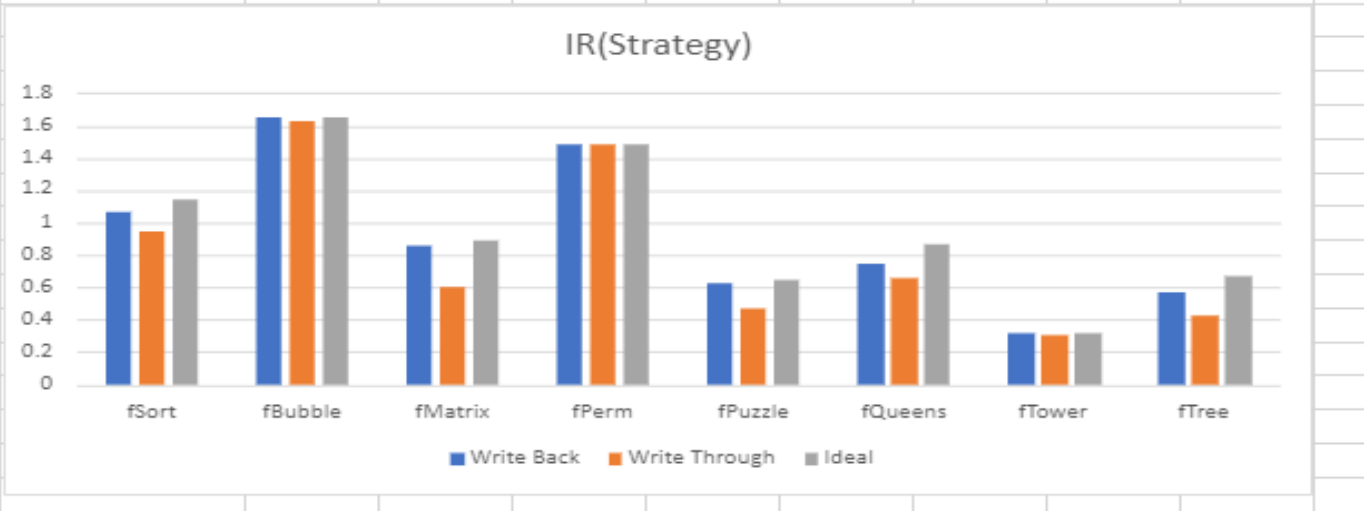
RmissDC(BLOCK_SIZE)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
4	12.1	0.36	28.84	0.04	60.21	14.75	7.93	34.54
8	7.29	0.19	17.39	0.02	32.16	10.68	4.63	30.14
16	4.46	0.1	11.31	0.01	18.81	7.71	5.55	28.36



Cresterea dimensiunii blocurilor din cache reprezinta o scadere a ratei de miss in cache-ul de date si o oarecare crestere a IR.
Strategia de scriere in memorie (Write Back sau Write t=Through) nu afecteaza nici IR, nici Rmiss in DC aproape deloc.

Ex. 6

IR(Strategy)	fSort	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fTower	fTree
Write Back	1.062	1.647	0.858	1.483	0.619	0.742	0.315	0.569
Write Through	0.945	1.633	0.598	1.481	0.466	0.655	0.299	0.423
Ideal	1.143	1.648	0.888	1.483	0.645	0.869	0.318	0.664

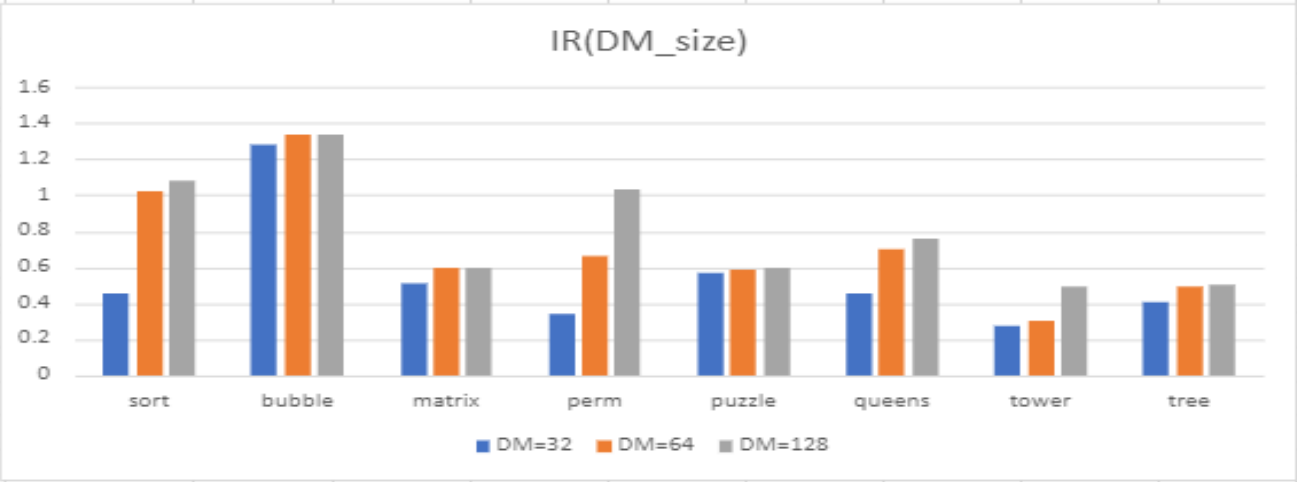


Write Through prezinta o mica scadere a IR in benchmark-urile care lucreaza cu date multe, in rest valorile sunt aproximativ egale. Cand se lucreaza cu date multe este de preferat sa folosim Write Back, nu Write Through.

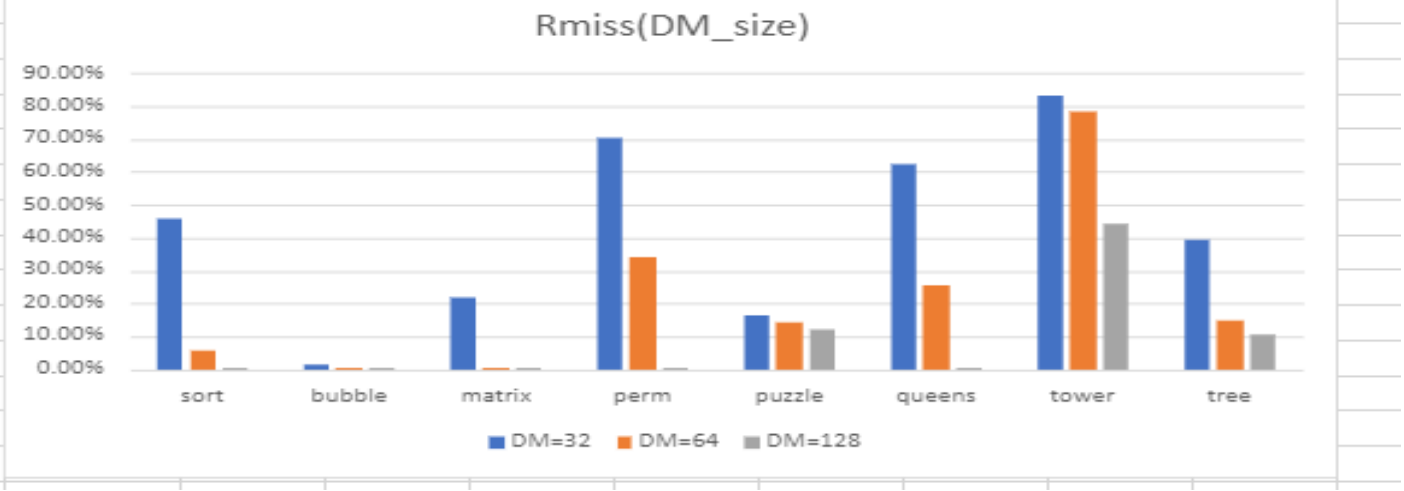
Lab 4 – VICTIM CACHE

Ex. 1

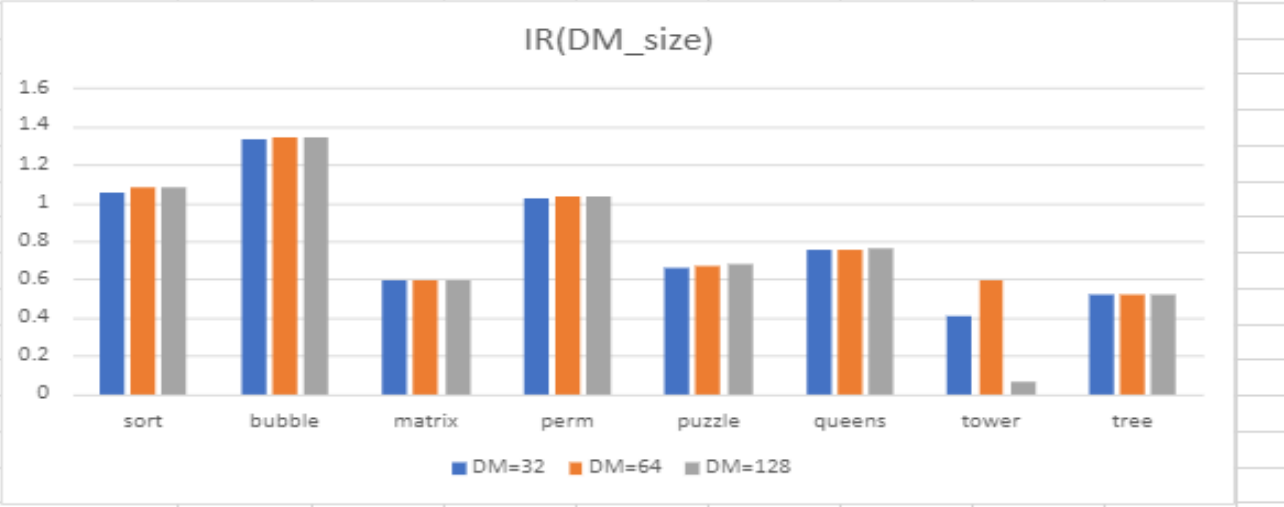
Fara Victim Cache								
IR	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	0.4499	1.2772	0.5139	0.3375	0.5622	0.4553	0.275	0.4033
DM=64	1.0194	1.34	0.5986	0.6595	0.5814	0.6989	0.301	0.4893
DM=128	1.0799	1.3407	0.5986	1.0355	0.5963	0.758	0.4952	0.5018



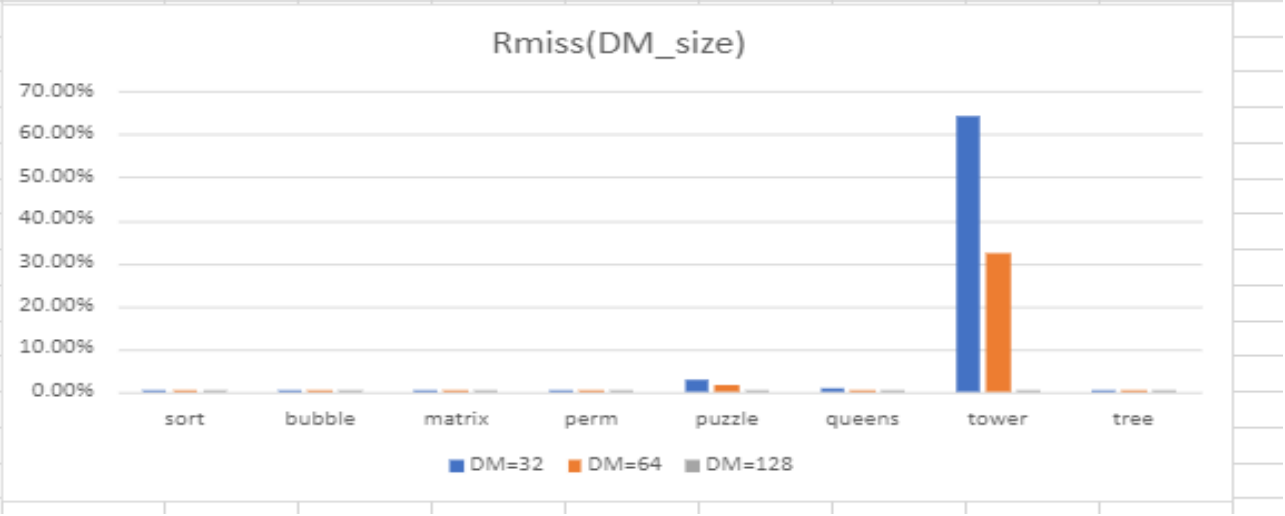
Fara Victim Cache								
Rmiss	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	45.69%	1.59%	21.57%	70.35%	16.58%	62.40%	83.41%	39.18%
DM=64	5.58%	0.06%	0.11%	33.86%	14.41%	25.66%	78.27%	14.99%
DM=128	0.27%	0.06%	0.09%	0.03%	11.93%	0.39%	44.44%	10.29%



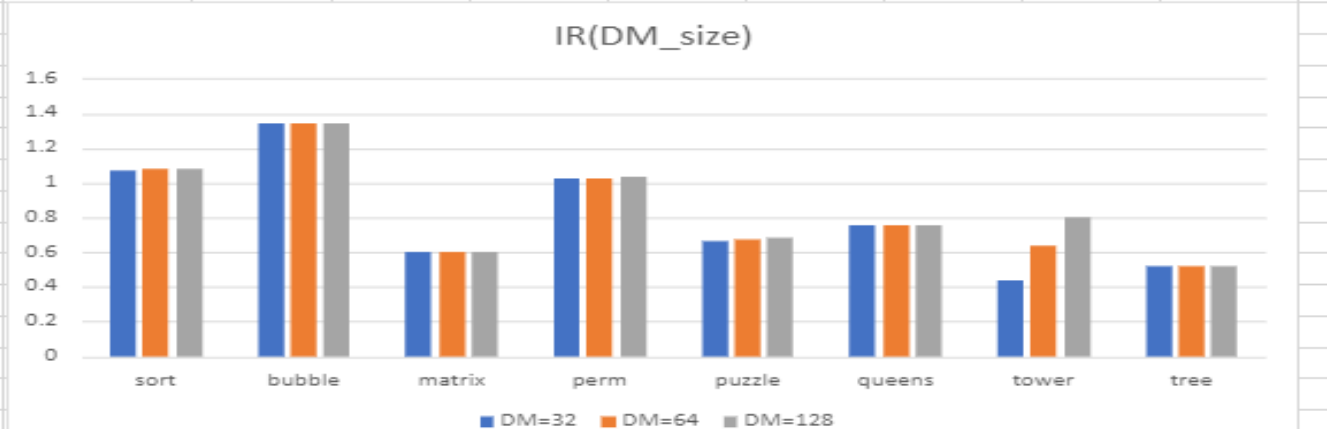
Victim Cache Simplu								
IR	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	1.0501	1.3337	0.5952	1.0204	0.6563	0.7564	0.4096	0.5158
DM=64	1.0795	1.3407	0.5987	1.0354	0.6694	0.7573	0.5914	0.5198
DM=128	1.0802	1.3407	0.5987	1.0355	0.6786	0.7593	0.06	0.52



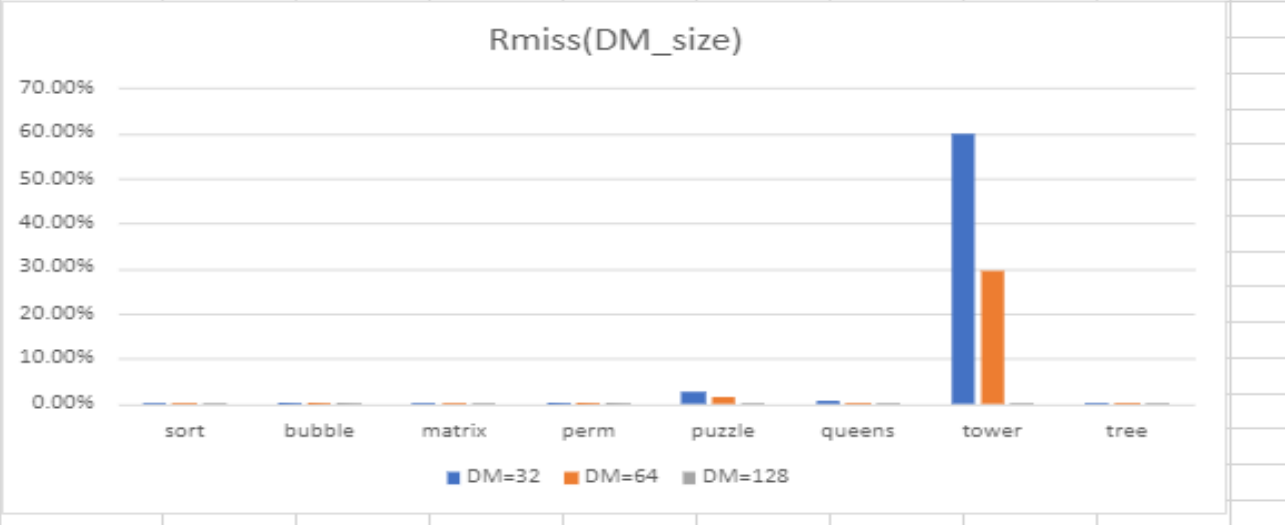
Victim Cache Simplu								
Rmiss	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	0.14%	0.03%	0.04%	0.02%	2.88%	0.49%	64.19%	0.06%
DM=64	0.14%	0.03%	0.04%	0.02%	1.36%	0.37%	32.13%	0.06%
DM=128	0.12%	0.03%	0.04%	0.02%	0.04%	0.05%	0.06%	0.06%



Selective Victim Cache								
IR	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	1.0729	1.3388	0.5984	1.0269	0.6625	0.7565	0.442	0.5186
DM=64	1.0795	1.3406	0.5987	1.0269	0.6714	0.7574	0.6402	0.5199
DM=128	1.0801	1.3407	0.5987	1.0355	0.681	0.7593	0.8015	0.52



Selective Victim Cache								
Rmiss	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	0.14%	0.03%	0.04%	0.02%	2.62%	0.49%	59.74%	0.07%
DM=64	0.13%	0.03%	0.04%	0.02%	1.54%	0.33%	29.46%	0.07%
DM=128	0.12%	0.03%	0.04%	0.02%	0.04%	0.05%	0.06%	0.06%



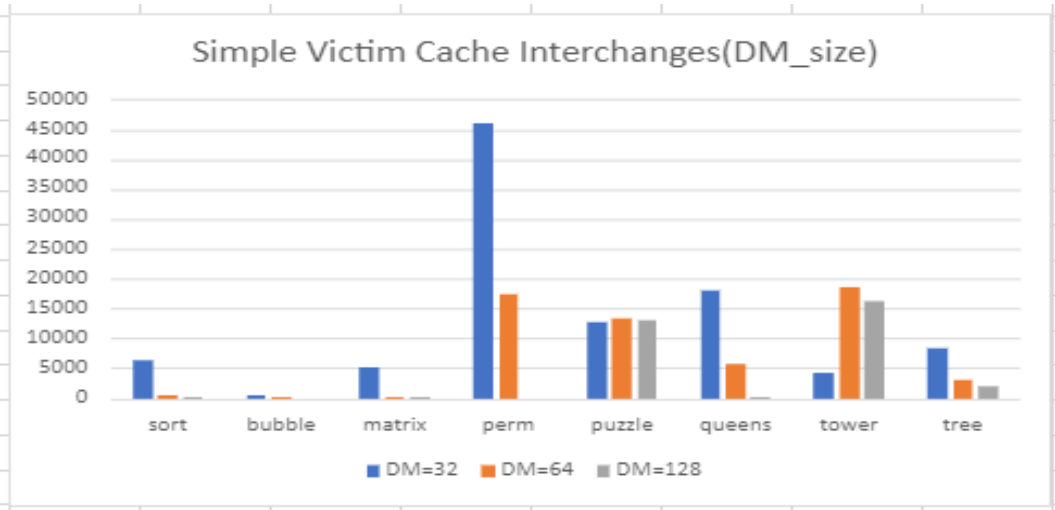
In cazul Fara Victim Cache, IR este mai bun cand dimensiunea cache-ului creste. Cu cat dimensiunea cache-ului este mai mare, cu atat mai mult se apropie valorile IR de la arhitectura Fara Victim Cache de valorile IR ale arhitecturii cu Victim Cache(atat Simplu, cat si Selective).
Exceptand benchmark-ul Tower, Victim Cache Simplu si Selective Victim Cache au un impact foarte asemanator asupra Instruction Rate-ului.

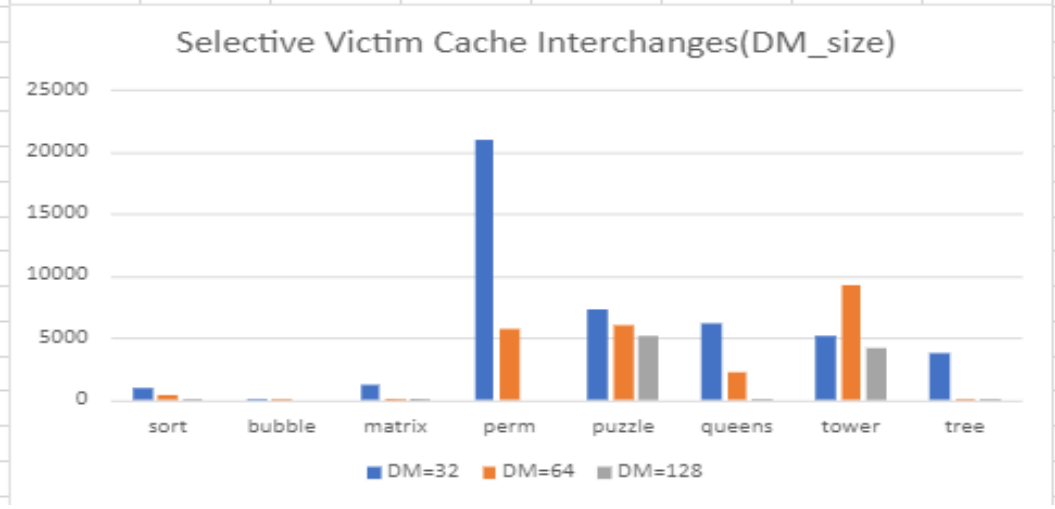
Prezenta Victim Cache-ului (atat Simplu, cat si Selective) are un impact semnificativ asupra Ratei de Miss, aceasta fiind mult mai mica fata de cea a unei arhitecturi Fara Victim Cache.

Ex. 2

Victim Cache Simplu								
Interchanges	**sort**	**bubble**	**matrix**	**perm**	**puzzle**	**queens**	**tower**	**tree**
DM=32	6241	399	5049	46135	12685	17971	4224	8477
DM=64	459	2	5	17321	13197	5746	18483	2992
DM=128	2	0	2	0	12992	56	16374	1996

Selective Victim Cache								
Interchanges	**sort**	**bubble**	**matrix**	**perm**	**puzzle**	**queens**	**tower**	**tree**
DM=32	987	7	1256	21006	7307	6218	5156	3680
DM=64	340	9	11	5682	6006	2137	9223	18
DM=128	5	0	2	0	5126	25	4119	13





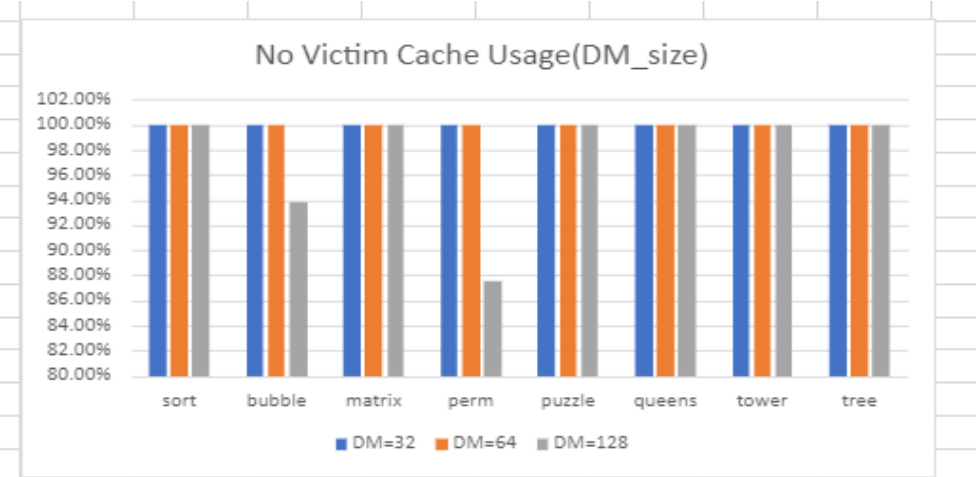
Dupa cum ne asteptam, numarul de interschimbari la selective Victim Cache este mai mic ca la Victim Cache simplu.

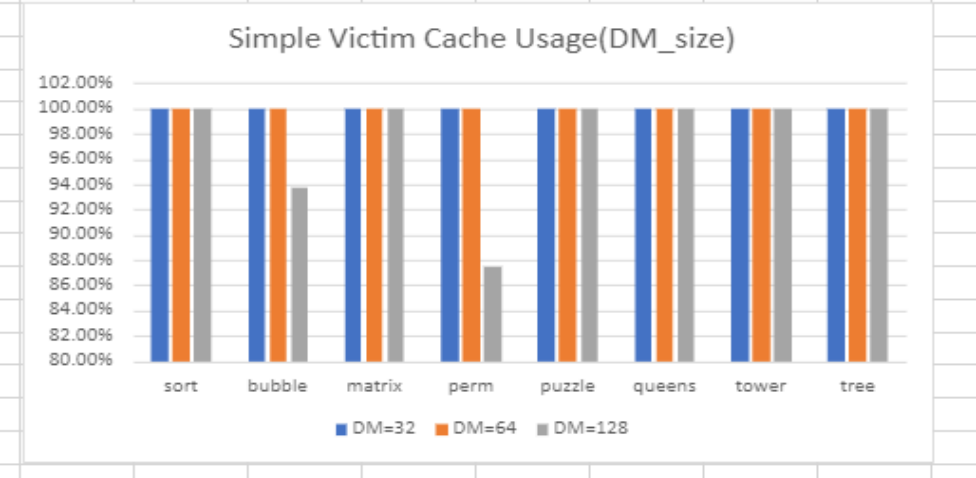
Ex. 3

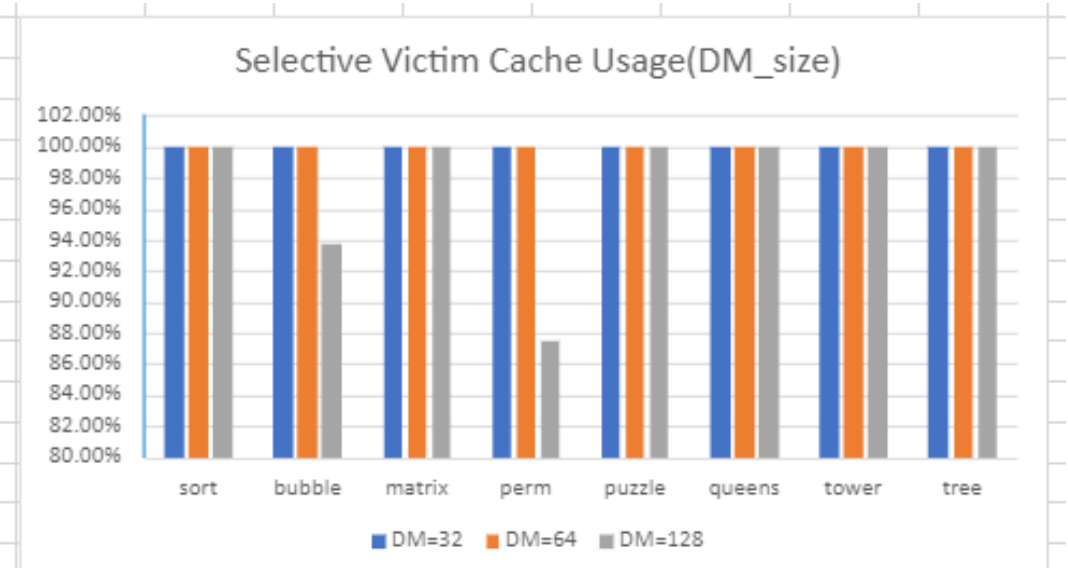
Fara Victim Cache								
Usage	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=64	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=128	100.00%	93.75%	100.00%	87.50%	100.00%	100.00%	100.00%	100.00%

Victim Cache Simplu								
Usage	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=64	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=128	100.00%	93.75%	100.00%	87.50%	100.00%	100.00%	100.00%	100.00%

Selective Victim Cache								
Usage	sort	bubble	matrix	perm	puzzle	queens	tower	tree
DM=32	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=64	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
DM=128	100.00%	93.75%	100.00%	87.50%	100.00%	100.00%	100.00%	100.00%







Dupa cum se poate observa, prezenta sau absenta Victim Cache-ului nu are un impact asupra Usage-ului.

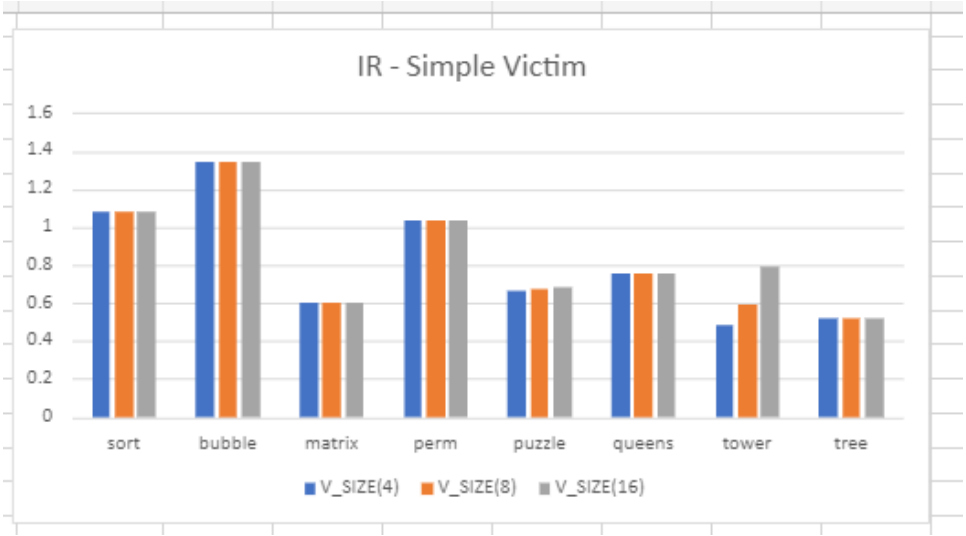
Ex. 4

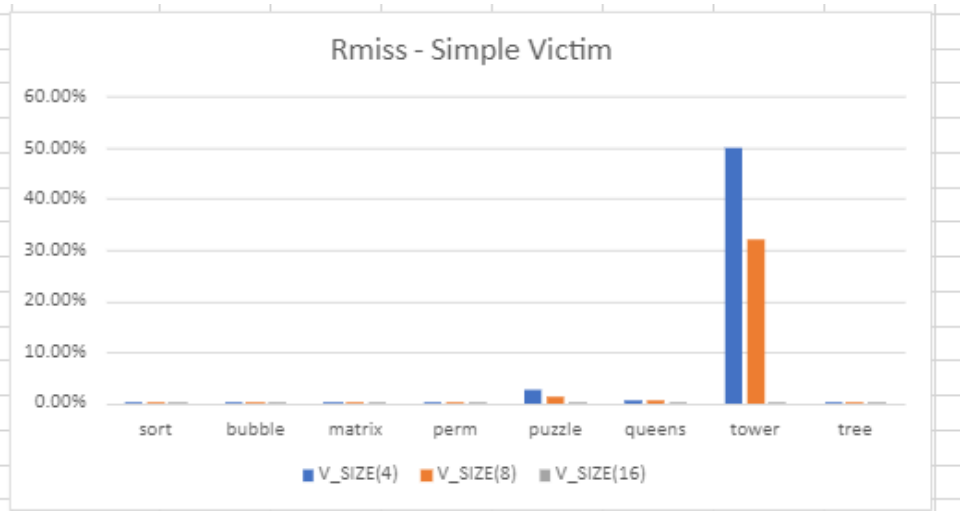
Instruction Cache - Simple Victim								
IR	sort	bubble	matrix	perm	puzzle	queens	tower	tree
V_SIZE(4)	1.0794	1.3405	0.5987	1.0354	0.6612	0.7566	0.4795	0.5198
V_SIZE(8)	1.0795	1.3407	0.5987	1.0354	0.6694	0.7573	0.5914	0.5198
V_SIZE(16)	1.0795	1.3407	0.5987	1.0354	0.6782	0.759	0.7922	0.5198

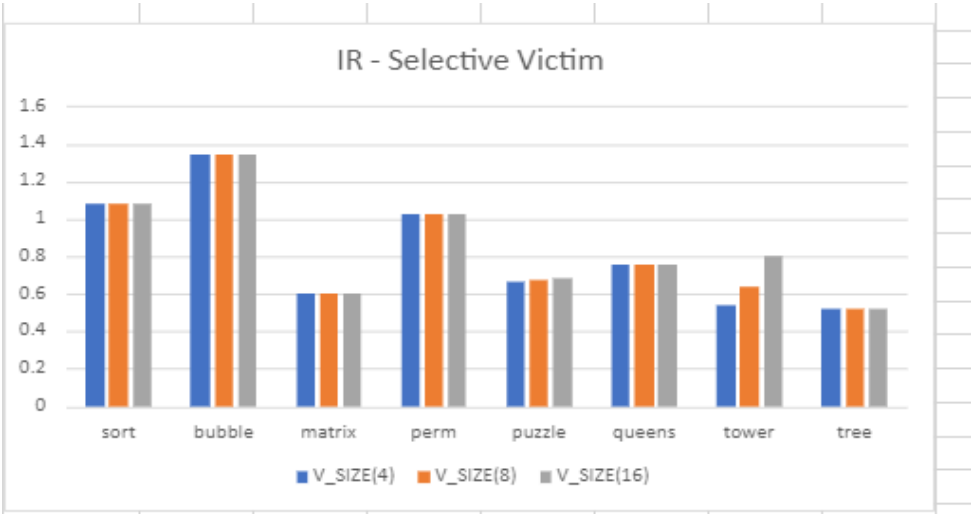
R miss	sort	bubble	matrix	perm	puzzle	queens	tower	tree
V_SIZE(4)	0.14%	0.03%	0.04%	0.02%	2.52%	0.49%	49.94%	0.07%
V_SIZE(8)	0.13%	0.03%	0.04%	0.02%	1.36%	0.37%	32.13%	0.06%
V_SIZE(16)	0.12%	0.03%	0.04%	0.02%	0.04%	0.05%	0.06%	0.06%

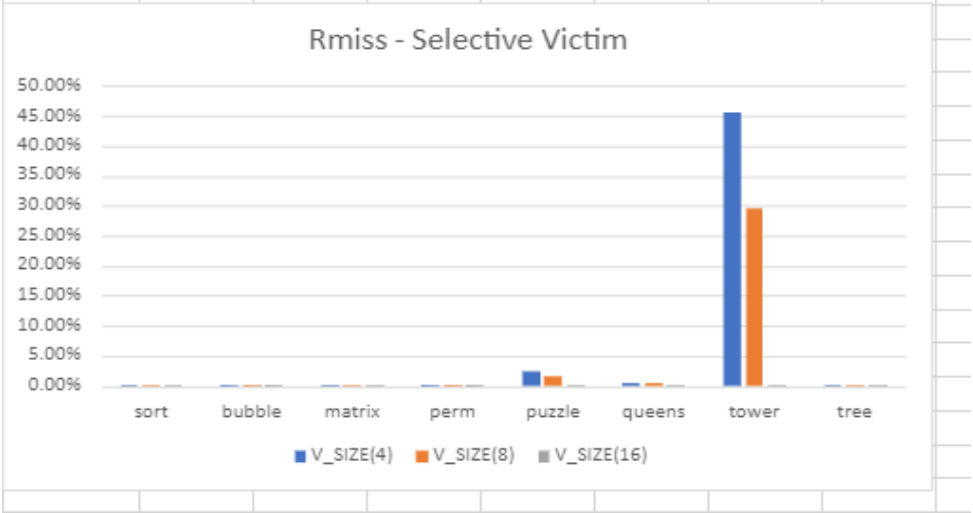
Instruction Cache - Selective Victim								
IR	sort	bubble	matrix	perm	puzzle	queens	tower	tree
V_SIZE(4)	1.0792	1.3405	0.5987	1.0269	0.6635	0.7564	0.538	0.5199
V_SIZE(8)	1.0795	1.3406	0.5987	1.0269	0.6714	0.7574	0.6401	0.5199
V_SIZE(16)	1.0795	1.3406	0.5987	1.0269	0.6808	0.759	0.7976	0.5199

R miss	sort	bubble	matrix	perm	puzzle	queens	tower	tree
V_SIZE(4)	0.15%	0.03%	0.05%	0.02%	2.56%	0.53%	45.49%	0.09%
V_SIZE(8)	0.13%	0.03%	0.04%	0.02%	1.54%	0.33%	29.46%	0.07%
V_SIZE(16)	0.12%	0.03%	0.04%	0.02%	0.04%	0.05%	0.06%	0.06%


IR - Simple Victim
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
sort
bubble
matrix
perm
puzzle
queens
tower
tree
V_SIZE(4)
V_SIZE(8)
V_SIZE(16)


Rmiss - Simple Victim
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
sort
bubble
matrix
perm
puzzle
queens
tower
tree
V_SIZE(4)
V_SIZE(8)
V_SIZE(16)



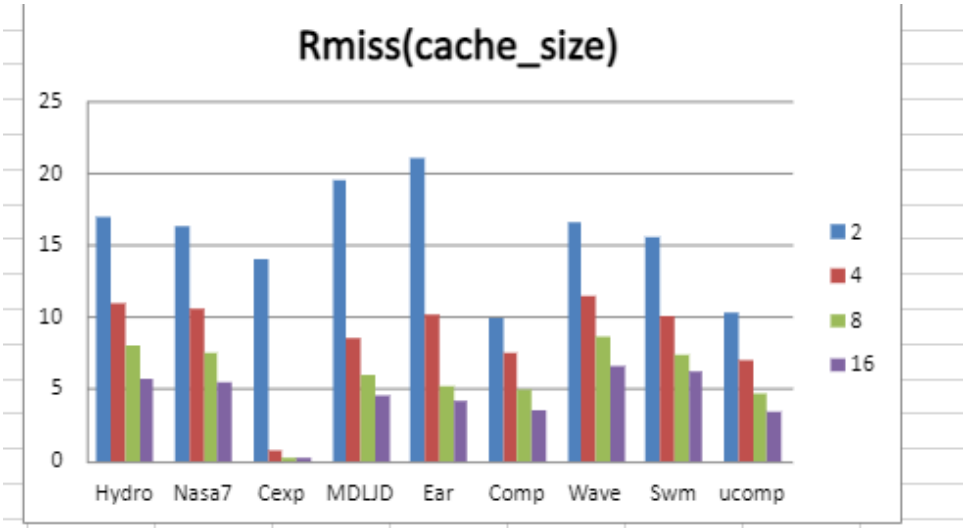


Singura diferenta care se poate observa este ca la Selective Victim Cache rata de miss este putin mai mica ca la Simple Victim Cache.

LAB 5 – SMPCACHE

Ex. 1

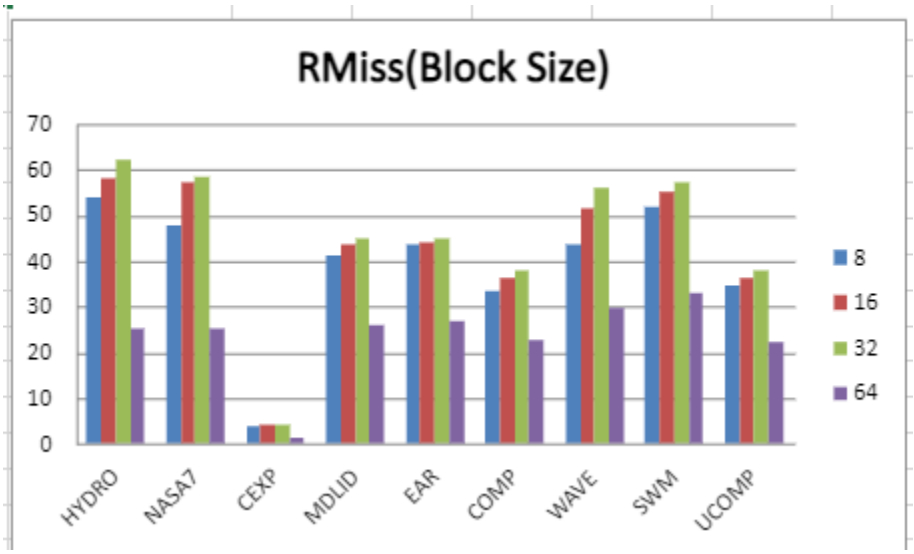
	Rmiss(cache_size)								
cache_size	**Hydro**	**Nasa7**	**Cexp**	**MDLJD**	**Ear**	**Comp**	**Wave**	**Swm**	**ucomp**
2	16.972	16.28	14.05	19.565	21.044	9.9445	16.603	15.535	10.245
4	11.001	10.566	0.7	8.465	10.211	7.4485	11.409	10	6.9887
8	8.0395	7.4933	0.235	5.94	5.1432	4.9525	8.6373	7.375	4.6469
16	5.6888	5.4987	0.215	4.595	4.107	3.5658	6.6239	6.22	3.4029



Cu cat dimensiunea cache-ului este mai mare, rata de miss scade considerabil.

Ex. 2

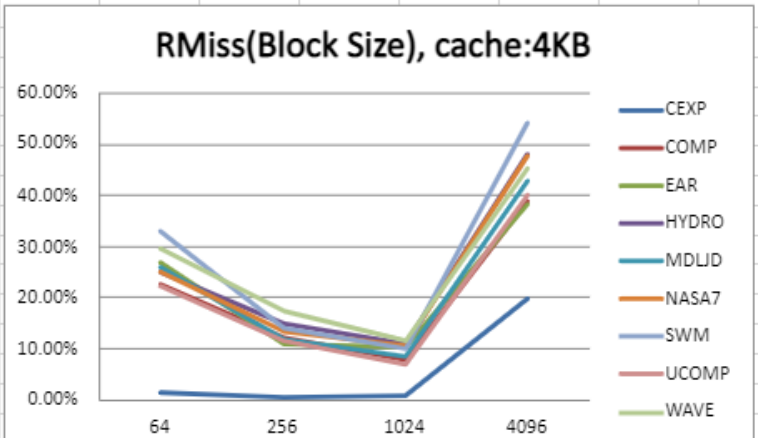
RMiss(BlockSize) %	HYDRO	NASA7	CEXP	MDLID	EAR	COMP	WAVE	SWM	UCOMP
8	53.97	47.87	3.81	41.32	43.5	33.51	43.79	51.99	34.61
16	58.11	57.25	4.14	43.64	44.23	36.13	51.41	55.34	36.15
32	62.06	58.6	4.22	44.82	44.93	37.87	55.96	57.36	37.94
64	25.15	25.07	1.31	25.94	26.92	22.7	29.73	33.04	22.24



Se observa ca rata de miss este in crestere pana la dimensiunea de 32, apoi scade brusc cand se face trecerea de la 32 la 64.

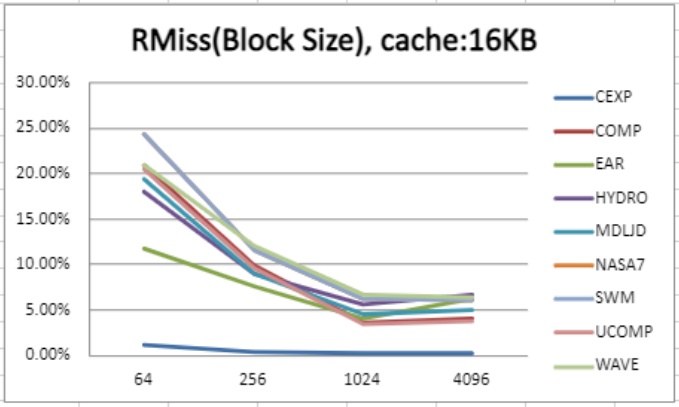
Ex. 3

Miss rate (block_size), Cache_size = 4KB									
	CEXP	COMP	EAR	HYDRO	MDLJD	NASA7	SWM	UCOMP	WAVE
64	1.31%	22.70%	26.92%	25.15%	25.94%	25.07%	33.04%	22.25%	29.73%
256	0.55%	12.01%	10.80%	14.81%	11.88%	13.37%	14.07%	11.49%	17.51%
1024	0.70%	7.74%	10.22%	11.00%	8.47%	10.57%	10.00%	6.99%	11.41%
4096	19.72%	38.99%	38.32%	47.96%	42.82%	47.87%	54.15%	39.96%	45.38%



Miss rate (block_size), Cache_size = 16KB

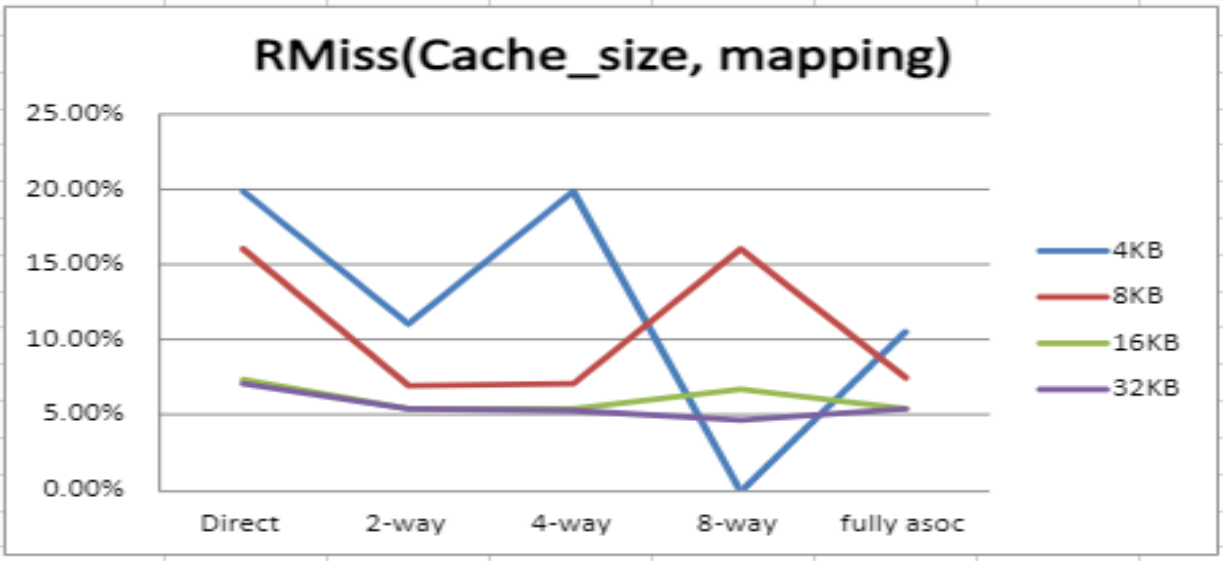
	CEXP	COMP	EAR	HYDRO	MDLJD	NASA7	SWM	UCOMP	WAVE
64	1.18%	20.96%	11.81%	18.05%	19.50%	24.40%	24.40%	20.53%	21.01%
256	0.46%	9.90%	7.63%	9.03%	8.97%	11.64%	11.64%	9.55%	11.99%
1024	0.22%	3.57%	4.11%	5.69%	4.60%	6.22%	6.22%	3.40%	6.62%
4096	0.19%	4.04%	6.27%	6.72%	5.07%	6.05%	6.05%	3.70%	6.45%



Rata de miss este in scadere pana la dimensiunea de 1024B, iar la trecerea de la 1024 la 4096 se observa o crestere brusca a ratei de miss, la cache-ul cu dimensiunea de 4KB. La cel de 16KB cresterea nu se poate observa. Cu toate acestea, noi putem presupune ca pragul s-a mutat mai departe (in cuvinte stiintifice).

Ex. 4

Nasa7				
Rmiss(cache_size, mapping)				
	4KB	**8KB**	**16KB**	**32KB**
Direct	19.89%	16.12%	7.39%	7.06%
2-way	11.10%	6.95%	5.50%	5.39%
4-way	19.89%	7.11%	5.50%	5.33%
8-way	-	16.11%	6.79%	4.74%
fully asoc	10.57%	7.49%	5.50%	5.39%

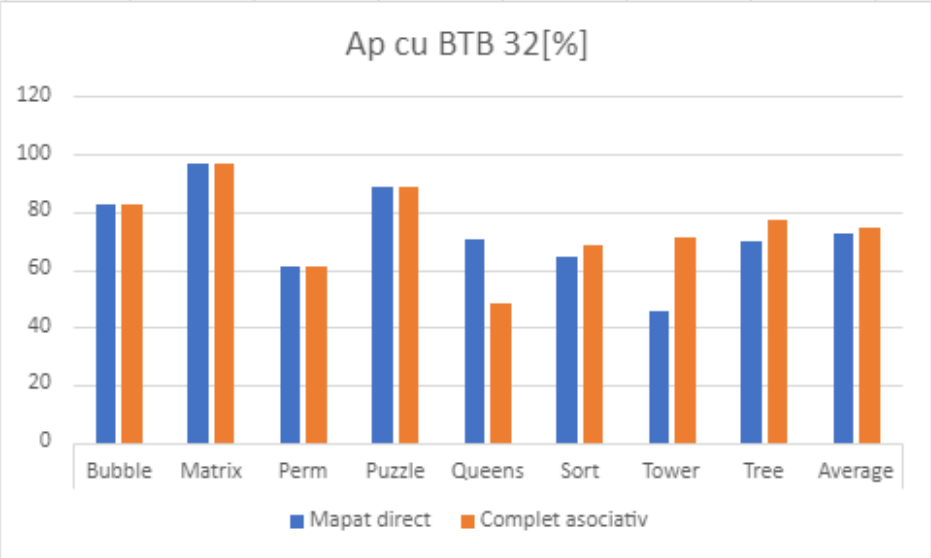


Se observa ca maparea directa are ce mai mare rata de miss, iar maparea 2-way set associative are cam aceeasi rata de miss ca maparea full associative.

Lab 6

Ex. A

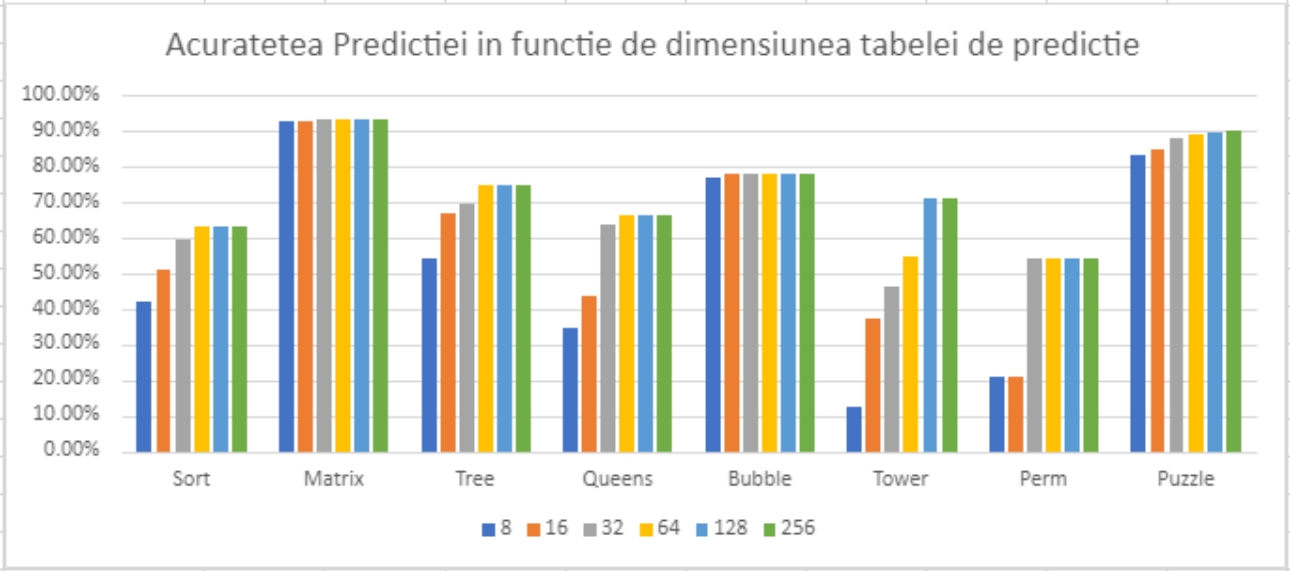
Ap cu btb 32 [%]									
	Bubble	Matrix	Perm	Puzzle	Queens	Sort	Tower	Tree	Average
Mapat dire	82.4	96.5	61.1	88.5	70.5	64.1	45.4	70.1	72.325
Complet a	82.9	96.5	61.1	88.9	48.2	68.7	70.9	76.9	74.2625



Se observa ca maparea complet asociativa este in medie mai buna ca cea directa.

Ex. B

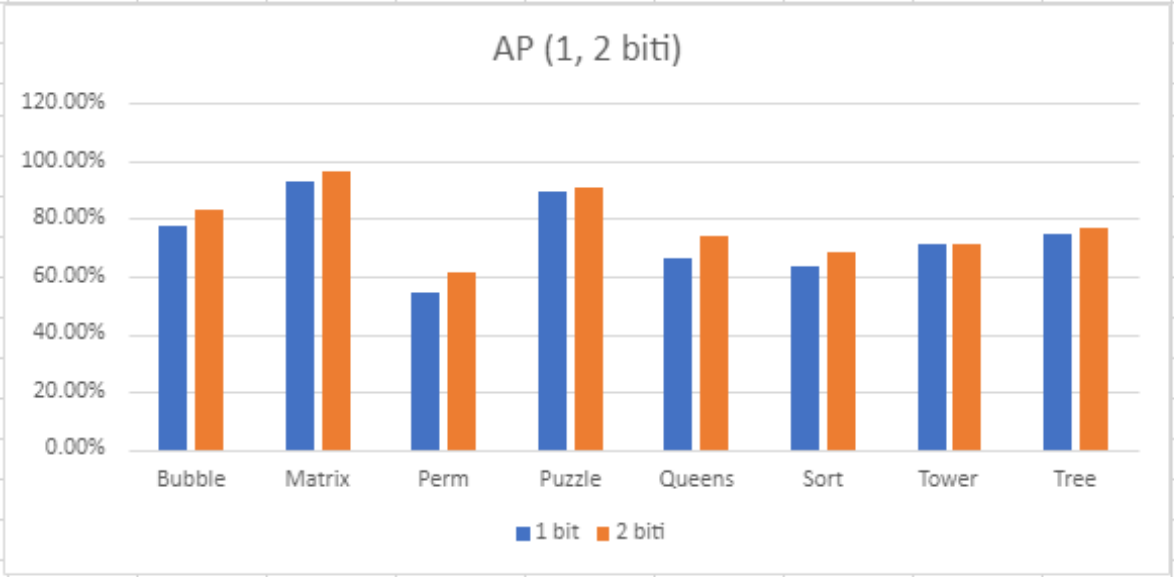
Mapat									
Nr Linii BTB	**Sort**	**Matrix**	**Tree**	**Queens**	**Bubble**	**Tower**	**Perm**	**Puzzle**	**Average**
8	42.07%	92.65%	54.40%	34.49%	76.83%	12.25%	20.95%	83.13%	52.10%
16	51.09%	92.87%	66.70%	43.47%	77.80%	37.35%	20.95%	84.73%	59.37%
32	59.73%	93.37%	69.76%	63.60%	77.80%	46.14%	54.44%	87.97%	69.10%
64	63.40%	93.30%	75.00%	66.40%	77.80%	55.00%	54.40%	89.20%	71.81%
128	63.40%	93.30%	75.00%	66.40%	77.80%	70.90%	54.40%	89.50%	73.84%
256	63.40%	93.30%	75.00%	66.40%	77.80%	70.90%	54.40%	90.00%	73.90%



Cu cat dimensiunea tabelei de predictie este mai mare, cu atat este mai buna acuratetea de predictie.

Ex. C

AP	Bubble	Matrix	Perm	Puzzle	Queens	Sort	Tower	Tree	Average
1 bit	77.80%	93.31%	54.44%	89.73%	66.38%	63.40%	70.98%	75.04%	73.89%
2 biti	82.96%	96.58%	61.16%	90.86%	73.73%	68.73%	70.99%	76.96%	77.75%

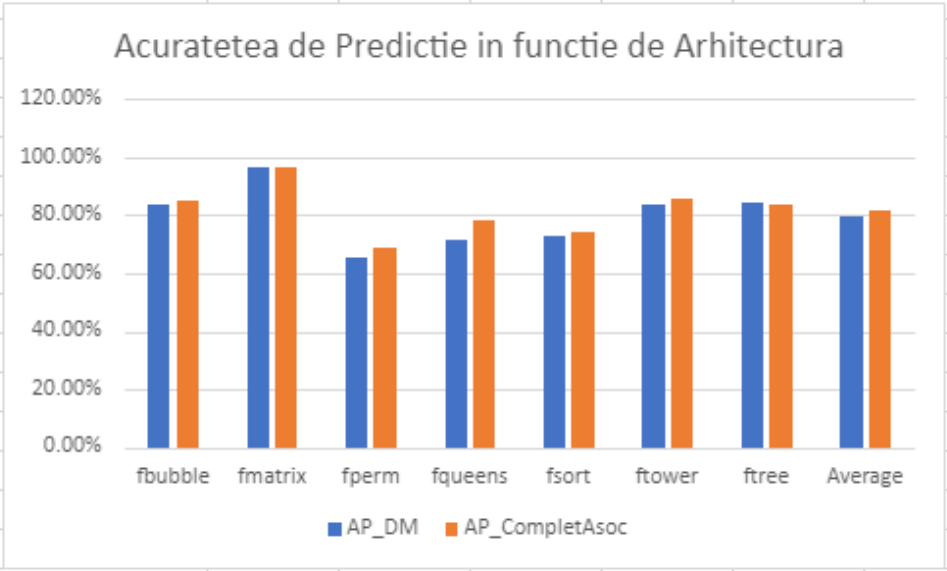


Acuratetea de predictie cand avem 2 biti este mai buna decat acuratetea de predictie cand avem 1 bit.
Cu toate acestea, diferenta intre cele 2 arhitecturi este minima.

Lab 7

Ex. 1

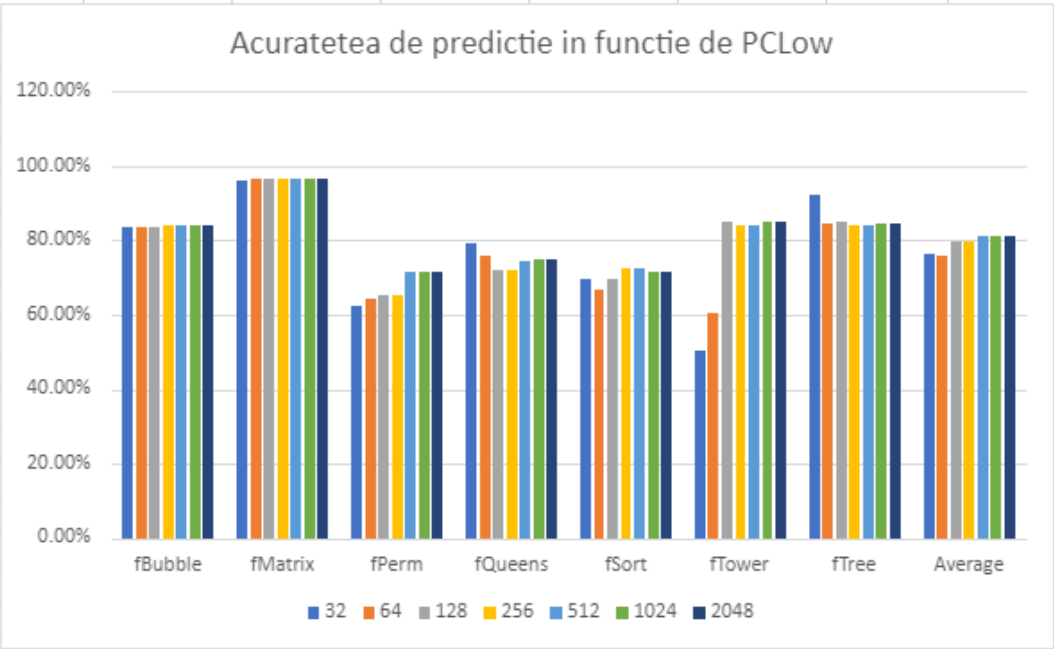
	fbubble	fmatrix	fperm	fqueens	fsort	ftower	ftree	Average
AP_DM	83.86%	96.68%	65.46%	71.86%	72.63%	84.02%	84.10%	79.80%
AP_CompletAsoc	85.29%	96.68%	68.49%	78.37%	74.15%	85.95%	84.04%	81.85%



Concluzie: Acuratetea de predictie este mai buna daca, la nivel de arhitectura hardware, avem cache complet asociativ.

Ex. 2

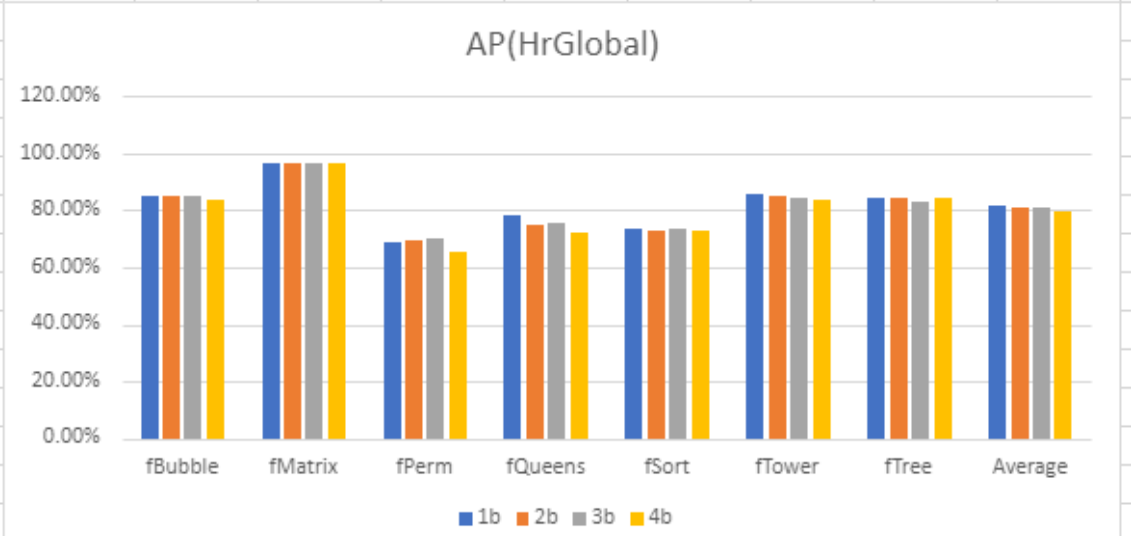
PCLow	fBubble	fMatrix	fPerm	fQueens	fSort	fTower	fTree	Average
32	83.71%	96.15%	62.19%	79.28%	69.58%	50.29%	92.38%	76.23%
64	83.71%	96.42%	64.54%	76.00%	66.53%	60.41%	84.81%	76.06%
128	83.71%	96.68%	65.46%	72.11%	69.74%	84.91%	84.98%	79.66%
256	83.86%	96.68%	65.46%	71.87%	72.64%	84.02%	84.10%	79.80%
512	83.86%	96.69%	71.33%	74.66%	72.71%	83.90%	84.21%	81.05%
1024	83.86%	96.69%	71.33%	74.77%	71.53%	84.94%	84.50%	81.09%
2048	83.86%	96.69%	71.33%	74.77%	71.53%	84.94%	84.50%	81.09%



In cazul a cele mai multe benchmark-uri, numarul mai mare al intrarilor in tabela creste acuratetea, insa, din simularile efectuate, diferenta asupra acuratetei este mica.

Ex. 3

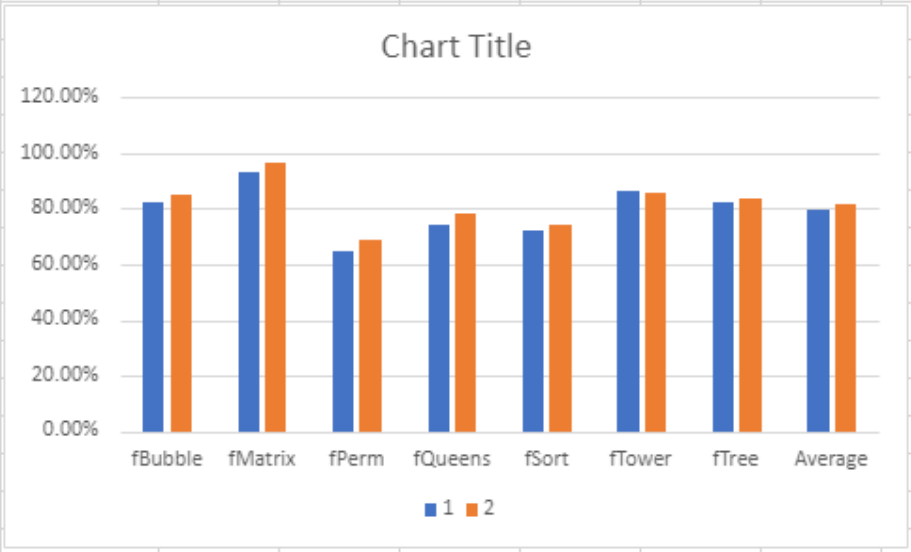
HR Global	fBubble	fMatrix	fPerm	fQueens	fSort	fTower	fTree	Average
1b	85.30%	96.69%	68.50%	78.38%	73.67%	85.95%	84.12%	0.818014
2b	85.34%	96.69%	69.17%	75.25%	72.74%	85.27%	84.15%	0.8123
3b	85.38%	96.69%	70.36%	75.44%	73.35%	84.25%	83.26%	0.812471
4b	83.86%	96.68%	65.46%	71.87%	72.64%	84.02%	84.10%	0.798043



In medie, cand HR Global = 1b, acuratetea este mai ridicata. Uneori marirea lui HR creste acuratetea, dar noi, la un moment dat, vom ajunge la un prag, dupa care acuratatea va incepe sa scade

Ex. 4

Acuratetea de predictie = fr(nr_biti)								
Bits number	fBubble	fMatrix	fPerm	fQueens	fSort	fTower	fTree	Average
1	82.45%	93.41%	64.90%	74.10%	71.87%	86.48%	82.32%	0.793614
2	85.30%	96.69%	68.50%	78.38%	74.15%	85.95%	84.04%	0.818586

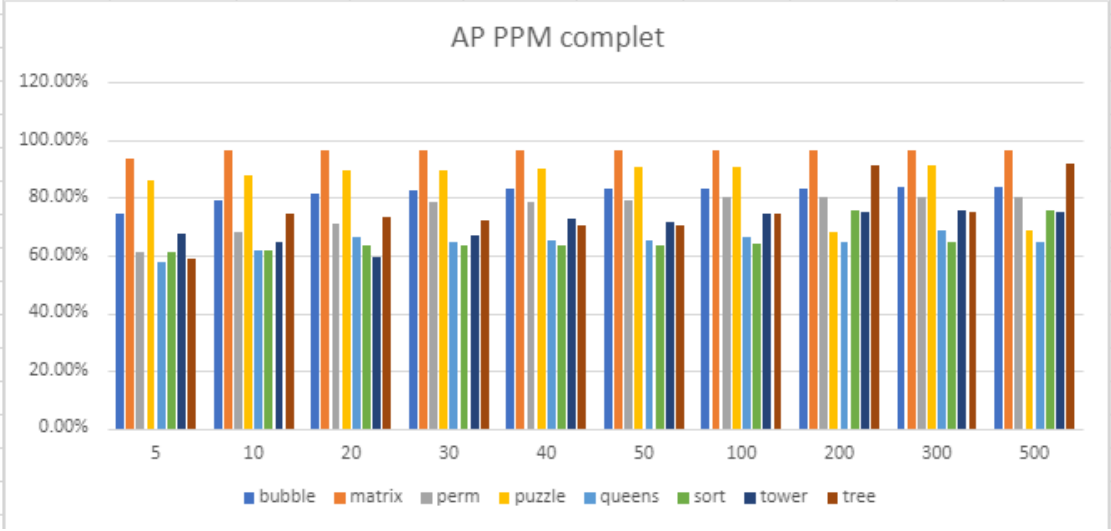


Dupa cum ne asteptam, avand 2 biti de predictie, acuratatea este mai buna, fata de 1 singur bit

Lab 8

Ex. 1

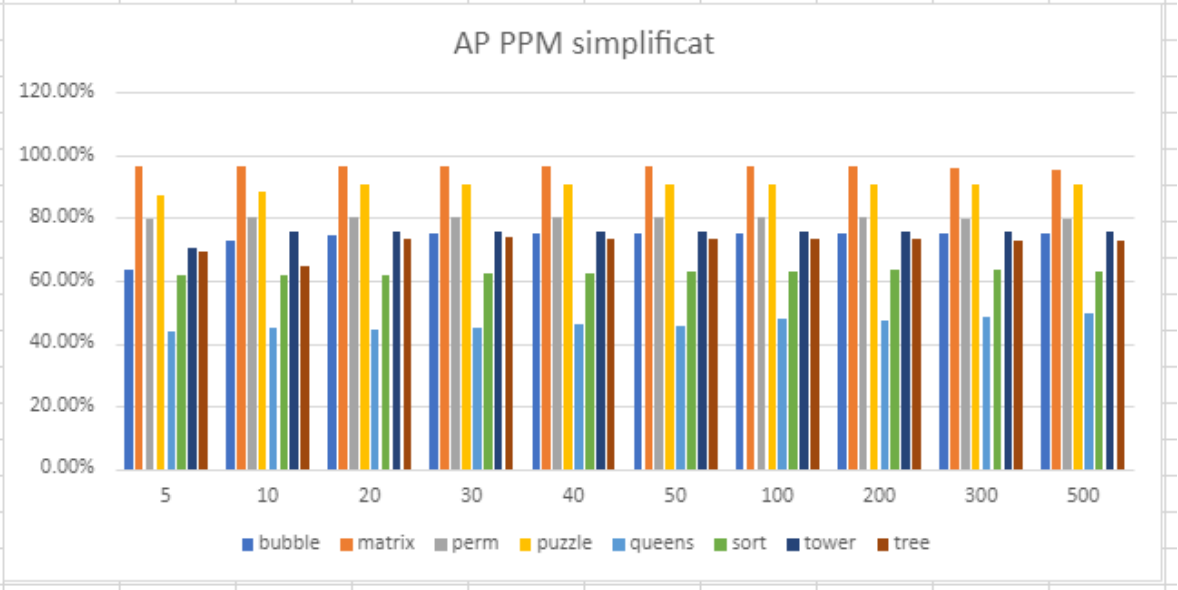
HRg	bubble	matrix	perm	puzzle	queens	sort	tower	tree	Average
5	74.44%	93.39%	61.32%	86.31%	57.74%	61.27%	67.48%	59.06%	70.13%
10	79.16%	96.33%	68.30%	88.05%	62.03%	61.87%	64.59%	74.39%	74.34%
20	81.40%	96.56%	70.73%	89.59%	66.35%	63.58%	59.51%	73.27%	75.12%
30	82.56%	96.56%	78.38%	89.67%	64.66%	63.36%	66.88%	72.21%	76.79%
40	83.04%	96.45%	78.68%	90.06%	65.45%	63.43%	72.89%	70.49%	77.56%
50	82.93%	96.45%	79.38%	90.46%	65.01%	63.75%	71.34%	70.68%	77.50%
100	83.21%	96.66%	80.17%	90.89%	66.10%	63.80%	74.74%	74.43%	78.75%
200	83.21%	96.68%	80.17%	68.04%	64.55%	75.85%	75.01%	91.15%	79.33%
300	83.66%	96.69%	80.17%	91.46%	68.44%	64.52%	75.85%	75.17%	79.50%
500	83.69%	96.69%	80.17%	68.96%	64.55%	75.85%	75.32%	91.61%	79.61%



Se observa ca la un HRg de 40-50 de biti, acuratetea de predictie incepe sa creasca tot mai putin, iar la valori mai mari precum 200-500 de biti cresterea este prea mica pentru a se merita sa implementam un HRg asa costisitor. Valoarea optima este in jur de 100 de biti.

Ex. 2

HRg	bubble	matrix	perm	puzzle	queens	sort	tower	tree	Average
5	63.55%	96.68%	79.70%	87.20%	43.51%	61.59%	70.68%	69.15%	71.51%
10	72.84%	96.67%	80.14%	88.31%	44.73%	61.60%	75.85%	64.64%	73.10%
20	74.60%	96.65%	80.16%	90.51%	44.49%	61.89%	75.84%	73.28%	74.68%
30	74.94%	96.63%	80.16%	90.70%	44.84%	62.58%	75.85%	73.84%	74.94%
40	74.99%	96.60%	80.16%	90.74%	46.19%	62.33%	75.81%	73.42%	75.03%
50	75.19%	96.58%	80.15%	90.77%	45.76%	62.62%	75.81%	73.40%	75.04%
100	75.24%	96.46%	80.12%	90.88%	47.67%	63.12%	75.76%	73.33%	75.32%
200	75.15%	96.23%	80.05%	90.86%	47.23%	63.29%	75.69%	73.18%	75.21%
300	75.03%	96.00%	79.98%	90.84%	48.60%	63.33%	75.58%	73.03%	75.30%
500	74.79%	95.54%	79.84%	90.79%	49.32%	62.99%	75.40%	72.73%	75.18%



Comparativ cu predictorul Markov complex putem observa ca nu exista salturi asa de mari in acuratetea de predictie a benchmark-urilor individuale, iar valoarea optima este la un HRg de 40 de biti, cresterea acuratetei de predictie fiind prea mica pentru a creste HRg-ul in continuare.

Lab 9

Ex. 1

Hidde Layer	HRG	fbubble	fmatrix	fperm	fpuzzle	fqueens	fsort	ftower	ftree	Avg
15	2	85.53	96.71	89.45	95.39	80.04	75.83	96.68	89.48	88.63875
	4	85.55	96.71	95.13	95.66	81.47	76.32	96.93	89.73	89.6875
	6	86	96.71	95.37	95.92	81.93	76.92	96.77	86.69	89.53875
	8	86.22	96.71	94.55	95.83	82.14	76.356667	96.58	90.07	89.80708
	10	86.23	96.71	94.23	95.67	83.55	76.356667	96.59	90.01	89.91833
30	2	85.46	96.7	89.44	95.32	80	76.46	96.78	89.48	88.705
	4	85.63	96.7	93.14	95.69	81.61	76.18	96.89	89.73	89.44625
	6	85.71	96.7	95.04	95.79	82.59	76.11	96.88	89.7	89.815
	8	86.32	96.7	94.23	95.86	82.8	76.17	96.6	89.9	89.8225
	10	86.21	96.7	94.28	95.86	83.9	76.23	96.58	90.08	89.98
50	2	85.65	96.69	88.65	95.26	79.93	75.84	96.76	89.4	88.5225
	4	85.5	96.7	92.35	95.63	82.02	76.18	97.03	89.75	89.395
	6	85.69	96.7	95.02	95.75	82.58	76.81	96.82	89.7	89.88375
	8	86.41	96.7	94.35	95.85	82.43	75.42	96.62	90.02	89.725
	10	86.41	96.7	94.16	95.86	82.91	76.0625	96.51	89.94	89.81906

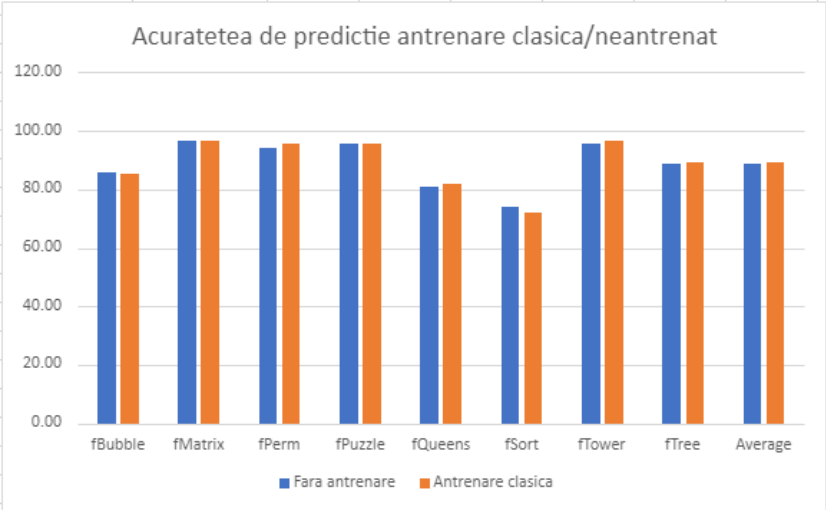
Ex. 2

Hidden Layer	15	Acuratetea								
HRG	Learning Step	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fSort	fTower	fTree	Average
0	0.125	84.84	96.71	89.79	93.51	75.23	71.14	89.91	86.57	85.96
1	0.250	85.28	96.71	89.86	94.56	78.96	71.66	95.84	88.14	87.63
2	0.375	85.57	96.71	88.78	95.04	79.67	72.96	96.15	88.26	87.89
3	0.500	85.78	96.70	88.99	95.12	79.95	73.16	96.35	88.62	88.08
4	0.625	85.68	96.71	93.01	95.37	80.50	73.21	96.55	88.94	88.75
5	0.750	85.73	96.71	92.14	95.44	80.43	73.10	96.60	89.08	88.65
6	0.875	85.82	96.71	94.20	95.43	81.53	74.60	96.14	89.02	89.18
7	1.000	85.88	96.71	94.22	95.30	81.40	74.38	95.66	87.27	88.85
8	1.125	85.85	96.70	95.20	95.32	81.12	72.19	95.18	88.83	88.80
9	1.250	85.87	96.70	95.42	95.61	81.72	73.87	94.84	88.35	89.05
10	1.375	85.90	96.71	95.18	95.48	81.88	73.35	89.80	86.69	88.12

Ex. 3

Hidden Layer de la ex 1	15
Learning step de la ex 2	6
HRG	8

	Acuratetea								
	fBubble	fMatrix	fPerm	fPuzzle	fQueens	fSort	fTower	fTree	Average
Fara antrenare	86.00	96.71	94.11	95.58	80.89	73.82	95.48	88.98	88.95
Antrenare clasica	85.27	96.71	95.54	95.66	81.78	71.95	96.64	89.33	89.11

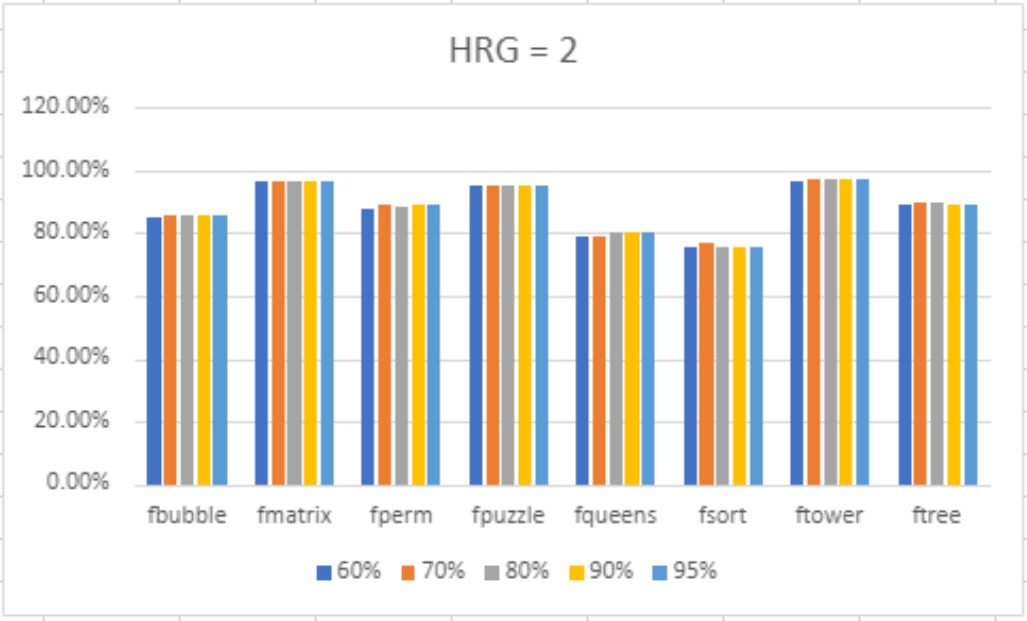


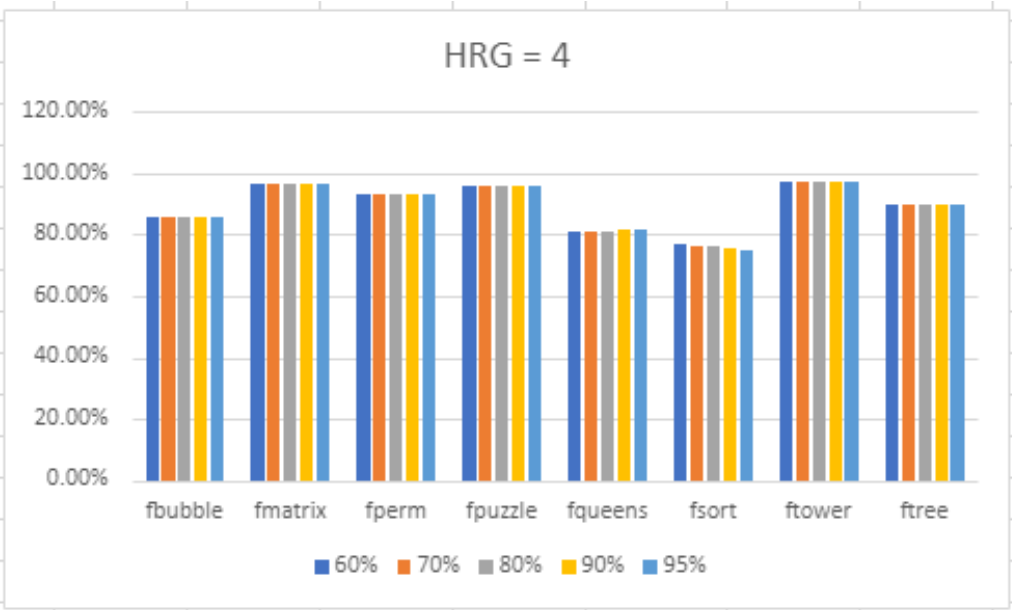
Acuratetea este mai buna daca avem antrenare clasica, insa diferenta fata de fara antrenare este mica.

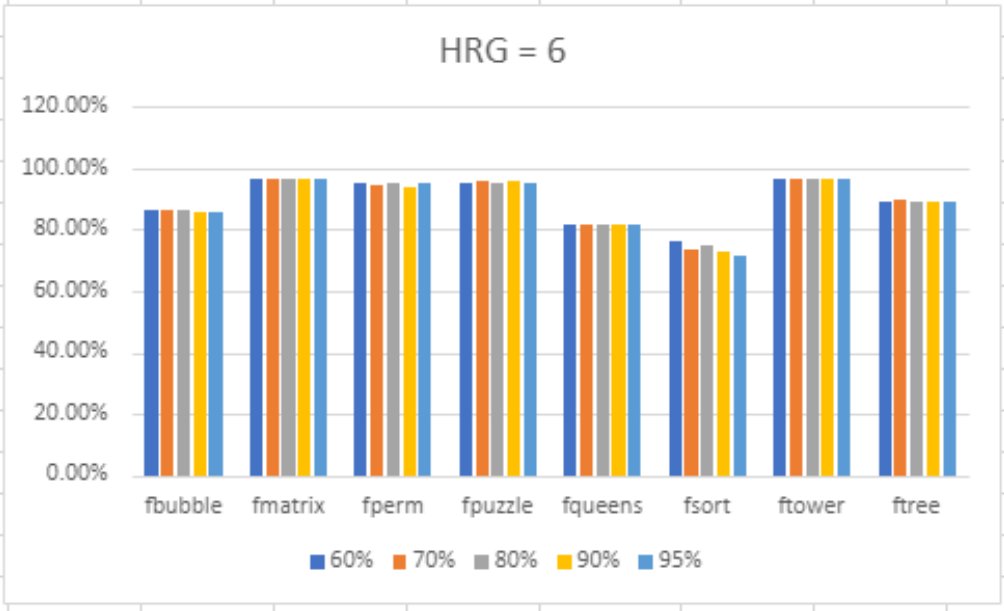
Ex. 4

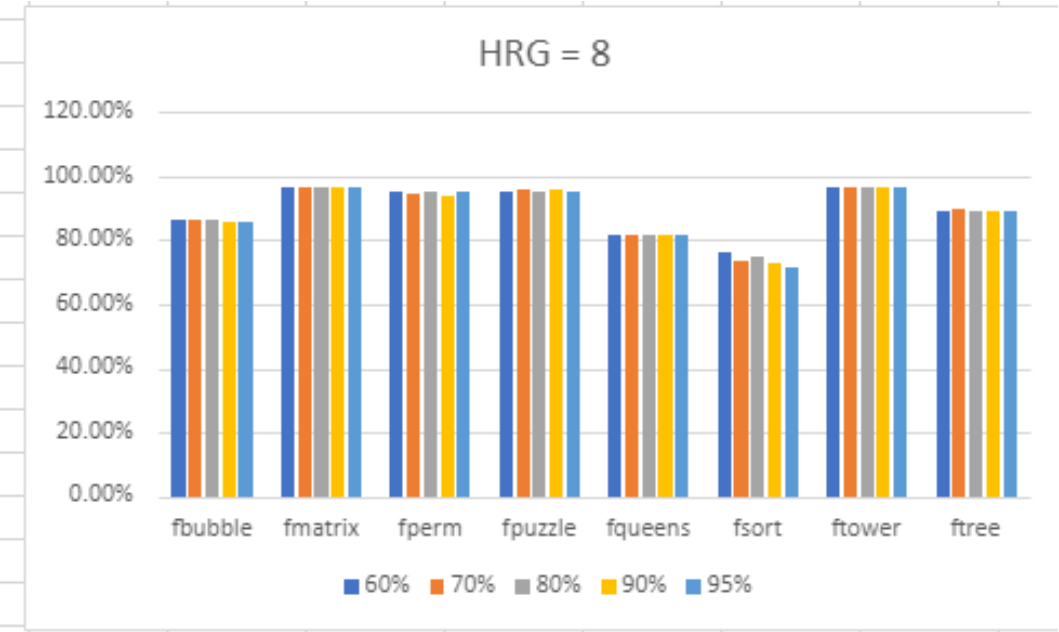
HRG	Filter	fbubble	fmatrix	fperm	fpuzzle	fqueens	fsort	ftower	ftree	Avg
2	60%	84.74%	96.71%	87.62%	95.15%	79.18%	75.59%	96.58%	89.05%	88.08%
	70%	85.48%	96.71%	89.10%	95.31%	79.25%	76.68%	96.95%	89.65%	88.64%
	80%	85.48%	96.71%	88.64%	95.33%	80.10%	75.34%	96.95%	89.65%	88.53%
	90%	85.48%	96.71%	89.02%	95.33%	80.26%	75.75%	96.95%	89.17%	88.58%
	95%	85.48%	96.71%	89.02%	95.18%	80.26%	75.75%	96.98%	89.35%	88.59%
4	60%	85.42%	96.71%	93.16%	95.70%	81.17%	76.61%	97.01%	89.77%	89.44%
	70%	85.53%	96.71%	92.92%	95.59%	81.30%	76.40%	97.01%	89.81%	89.41%
	80%	85.45%	96.71%	93.14%	95.62%	81.25%	76.43%	97.01%	89.63%	89.41%
	90%	85.45%	96.71%	93.13%	95.59%	81.49%	75.72%	97.01%	89.70%	89.35%
	95%	85.45%	96.71%	93.13%	95.60%	81.51%	75.25%	97.04%	89.70%	89.30%
6	60%	86.27%	96.71%	95.19%	95.47%	81.66%	76.12%	96.63%	89.22%	89.66%
	70%	86.27%	96.71%	94.23%	95.60%	81.58%	73.86%	96.29%	89.59%	89.27%
	80%	86.24%	96.71%	95.35%	95.38%	81.79%	75.16%	96.32%	89.17%	89.52%
	90%	85.64%	96.71%	93.86%	95.59%	81.54%	72.73%	96.41%	89.15%	88.95%
	95%	85.59%	96.71%	95.05%	95.48%	81.41%	71.55%	96.50%	89.00%	88.91%
8	60%	86.28%	96.71%	94.89%	95.77%	81.69%	75.26%	95.89%	89.75%	89.53%
	70%	86.58%	96.71%	95.60%	95.86%	81.88%	75.26%	96.74%	89.77%	89.80%
	80%	86.48%	96.71%	95.70%	95.95%	81.50%	75.26%	96.33%	89.87%	89.73%
	90%	86.01%	96.71%	95.52%	95.44%	82.45%	75.26%	95.69%	89.39%	89.56%
	95%	85.27%	96.71%	95.54%	95.66%	81.78%	75.26%	96.36%	89.96%	89.57%
10	60%	86.55%	96.71%	94.53%	95.25%	82.25%	75.26%	96.74%	89.67%	89.62%
	70%	86.58%	96.71%	94.64%	95.82%	83.00%	75.26%	96.05%	89.55%	89.70%
	80%	86.66%	96.71%	94.23%	95.77%	82.69%	75.26%	96.71%	89.63%	89.71%
	90%	85.51%	96.71%	95.61%	95.48%	79.29%	75.26%	96.88%	89.46%	89.28%
	95%	85.47%	96.71%	95.57%	95.49%	82.36%	75.26%	96.99%	89.01%	89.61%

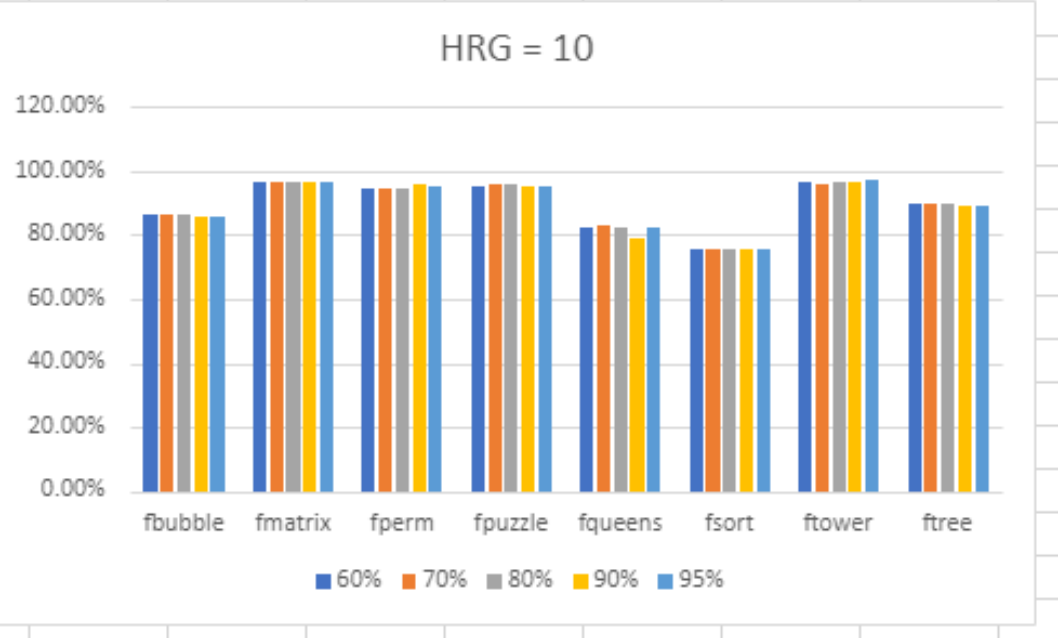
Procentul optim de filtrare a statisticilor este in jur de 60-70%.




HRG = 4
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
fbubble
fmatrix
fperm
fpuzzle
fqueens
fsort
ftower
ftree
60%
70%
80%
90%
95%


HRG = 6
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
fbubble
fmatrix
fperm
fpuzzle
fqueens
fsort
ftower
ftree
60%
70%
80%
90%
95%

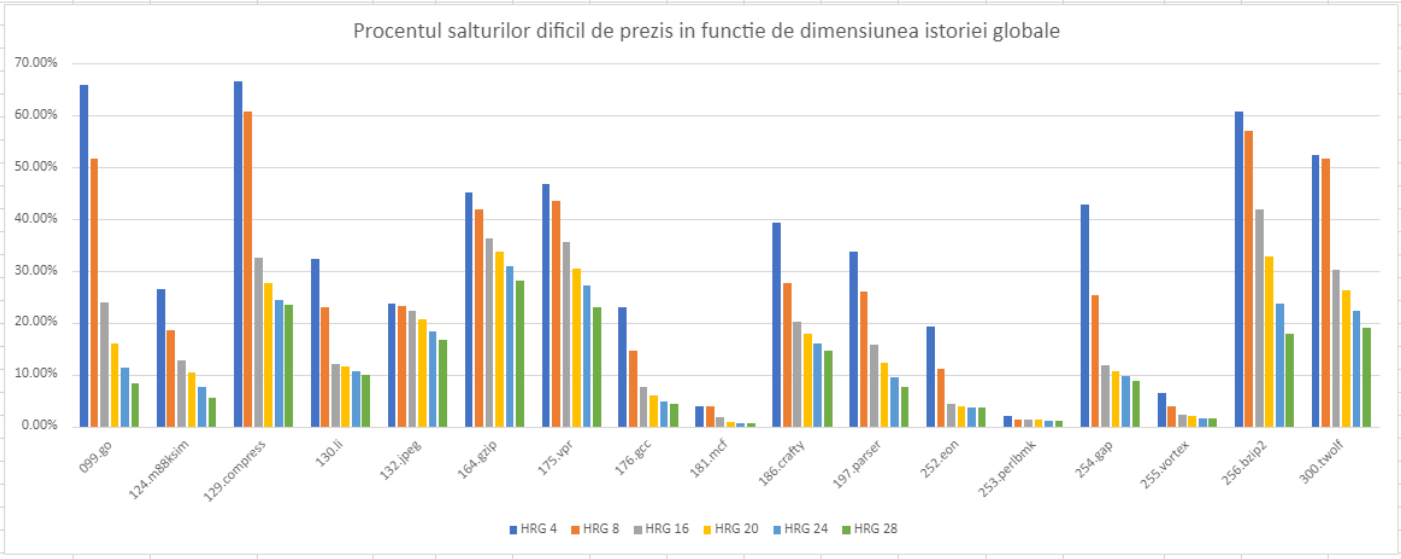

HRG = 8
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
fbubble
fmatrix
fperm
fpuzzle
fqueens
fsort
ftower
ftree
60%
70%
80%
90%
95%


HRG = 10
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
fbubble
fmatrix
fperm
fpuzzle
fqueens
fsort
ftower
ftree
60%
70%
80%
90%
95%

Lab 10

Ex. A

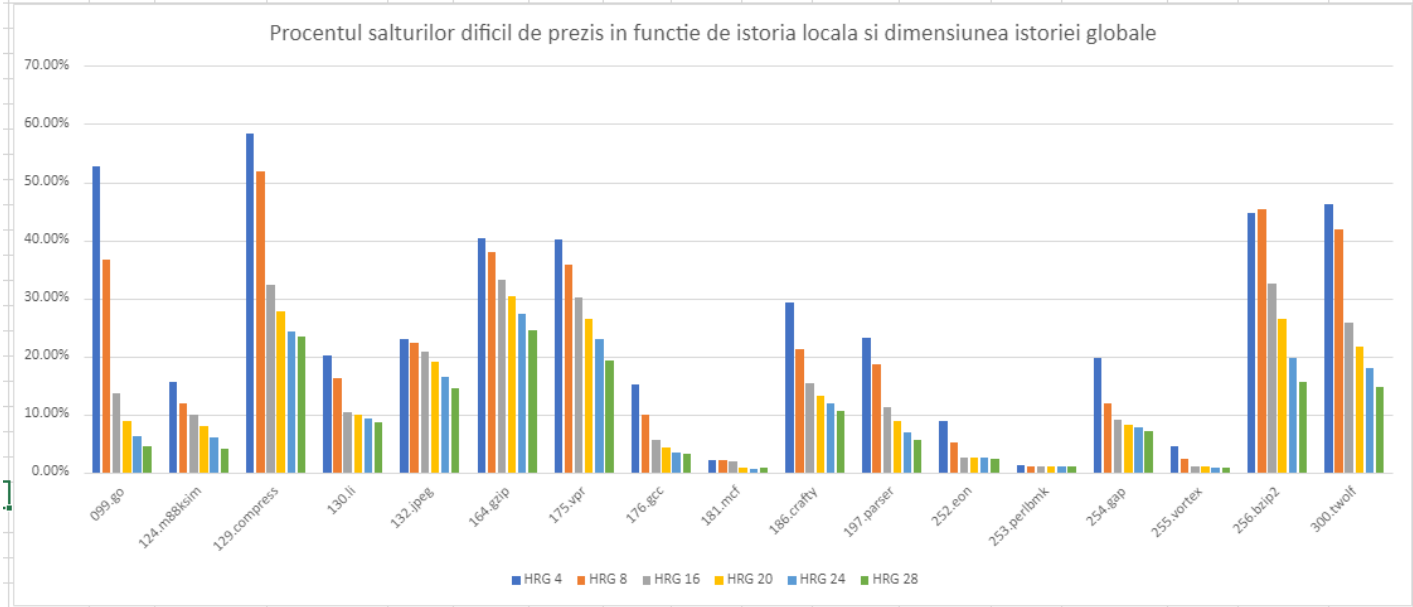
	099.go	124.m88ksim	129.compress	130.li	132.jpeg	164.gzip	175.vpr	176.gcc	181.mcf	186.crafty	197.parser	252.eon	253.perlbmk	254.gap	255.vortex	256.bzip2	300.twolf
HRG 4	65.88%	26.62%	66.50%	32.45%	23.79%	45.14%	46.82%	23.10%	3.87%	39.24%	33.72%	19.32%	2.05%	42.74%	6.44%	60.68%	52.34%
HRG 8	51.78%	18.47%	60.79%	22.94%	23.21%	41.88%	43.49%	14.66%	3.85%	27.67%	26.03%	11.23%	1.36%	25.41%	3.84%	56.95%	51.78%
HRG 16	23.96%	12.71%	32.63%	12.17%	22.33%	36.24%	35.66%	7.73%	1.90%	20.30%	15.80%	4.37%	1.23%	11.78%	2.19%	41.80%	30.20%
HRG 20	15.93%	10.33%	27.73%	11.50%	20.68%	33.80%	30.51%	6.06%	0.85%	17.79%	12.28%	3.78%	1.29%	10.67%	2.02%	32.91%	26.22%
HRG 24	11.31%	7.55%	24.31%	10.74%	18.30%	31.03%	27.13%	4.79%	0.71%	16.06%	9.58%	3.68%	1.20%	9.81%	1.64%	23.66%	22.33%
HRG 28	8.32%	5.56%	23.49%	10.06%	16.76%	28.17%	23.07%	4.44%	0.74%	14.60%	7.52%	3.56%	1.15%	8.82%	1.54%	17.79%	19.00%



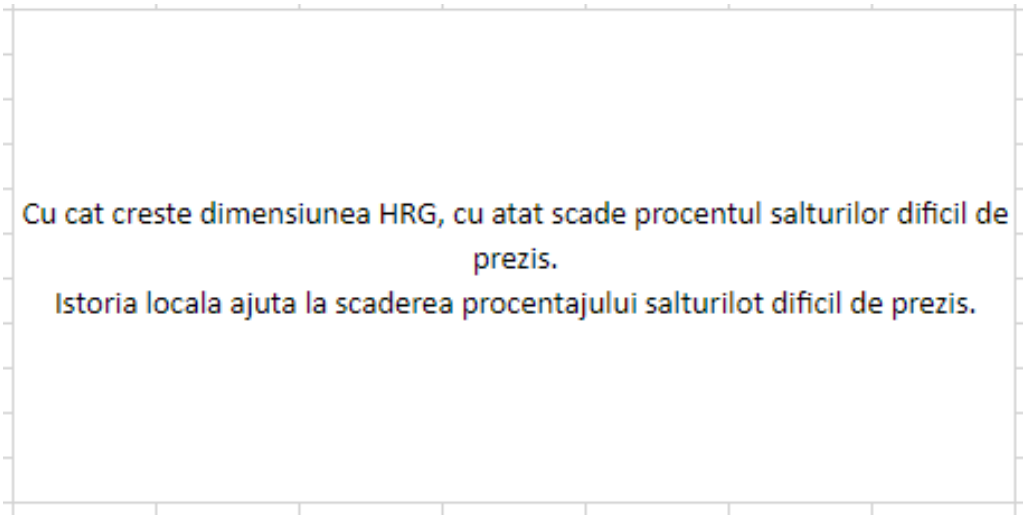
Ex. B

Istorie Locala = 4

	099.go	124.m88ksim	129.compress	130.li	132.jpeg	164.gzip	175.vpr	176.gcc	181.mcf	186.crafty	197.parser	252.eon	253.perlbmk	254.gap	255.vortex	256.bzip2	300.twolf
HRG 4	52.77%	15.49%	58.36%	20.15%	23.01%	40.39%	40.04%	15.14%	2.20%	29.34%	23.14%	8.82%	1.28%	19.62%	4.58%	44.74%	46.24%
HRG 8	36.68%	11.92%	51.77%	16.17%	22.31%	38.01%	35.84%	9.91%	2.13%	21.13%	18.61%	5.24%	1.09%	11.87%	2.25%	45.25%	41.80%
HRG 16	13.66%	9.93%	32.25%	10.43%	20.81%	33.10%	30.10%	5.53%	1.89%	15.26%	11.29%	2.49%	0.99%	9.10%	1.05%	32.44%	25.81%
HRG 20	8.88%	7.94%	27.73%	10.02%	18.99%	30.44%	26.40%	4.25%	0.83%	13.29%	8.82%	2.49%	1.05%	8.22%	0.97%	26.34%	21.65%
HRG 24	6.13%	5.96%	24.31%	9.23%	16.39%	27.40%	22.98%	3.37%	0.69%	11.81%	6.96%	2.46%	1.02%	7.76%	0.84%	19.73%	17.94%
HRG 28	4.40%	3.97%	23.49%	8.61%	14.47%	24.38%	19.19%	3.15%	0.72%	10.63%	5.61%	2.29%	1.02%	7.03%	0.82%	15.54%	14.66%



Ex. C


Cu cat creste dimensiunea HRG, cu atat scade procentul salturilor dificil de prezis.
Istoria locala ajuta la scaderea procentajului salturilot dificil de prezis.