Simulation experiments for hide-and-seek with different seeker distribution update strategies

Tirtharaj Dash

January 08, 2020

1 Preliminary Setup

We perform some simulation experiments for Hide-and-Seek with three different **seeker distribution update strategies**:

(1) **No update:** no update of the seeker distribution (leads to hide-and-seek with replacement results)

(2) **Uniform update:** open a box, distribute its probability mass to every other unopened boxes, make its probability 0 (hide-and-seek without replacement)

(3) **Hot-cold update:** open a box, check if it is hot (based on some threshold criteria), if it is hot, distribute its mass to its unopened neighbors; otherwise distribute its mass to every other unopened boxes, and make its probability 0 (hide-and-seek without replacement). We call a box i hot, if the performance of the box i, $perf(i) \geq \theta_h$.

To observe the expected number of misses the **hider distribution** is varied across four different **forms**:

(1) **Easy hider:** The hider distribution is easy. It translate to a single spike (degenerate distribution) such that there is absolute certainty that the hider will always hide there.

(2) **Not-so-easy hider:** The hider distibution is less easy. It translates to a distribution where there are some boxes which have masses and majority of boxes have zero probability mass. We can call it semi-degenerate hider.

(3) **Not-so-hard hider:** Here the hider distribution is not-uniform. It is a random distribution. Each box have a non-zero probability mass such that it is not a uniform distribution.

(4) **Hard hider:** The hider distribution is adversarial. This translates to a uniform hider distribution in which randomness is very high.

2 Base Experiments

Parameter setting The experiments are performed for number of boxes $n = \{1000, 2000, 3000\}$. The maximum hiding trials is set at 1000. We call it a **failure**, if the hider is not found within n searches by using the seeker distribution. The threshold temperature for deciding whether a box is hot is varied as $\theta_h = \{0.50, 0.80\}$. The neighborhood size (nbd) is fixed at 1 i.e. a box i has neighbors $i-1$ and $i+1$ except for the extreme cases like $i = 1$ and $i = n$ for which neighbors are $i+1$ and $i-1$ respectively. For simulation with not-so-easy hider, we have fixed the degeneracy to 25% i.e. a hider distribution in which 25% boxes have non-zero probability. For all ther experiments reported here, we define performance of a box by: $perf(i) = \frac{h_i}{\max(h_1,\ldots,h_n)}$, where $H = \{h_1, \ldots, h_n\}$ is the hider distribution.

Results The mean and standard deviations of misses are calculated only for successful runs i.e. the hider was found by the seeker within n look-ups. Otherwise, it was treated as a failure and this result was not included for statistics.

choiceH	choiceUpdS	SuccessRate	mean(misses)	sd(misses)
		$n = 1000$		
1	1	0.632	430.848	290.980
1	2	1.000	507.457	292.641
1	3	1.000	492.347	287.209
2	1	0.649	417.556	276.379
2	2	1.000	493.118	292.701
2	3	1.000	507.946	294.560
3	1	0.608	437.202	289.004
3	2	1.000	492.452	286.495
3	3	1.000	494.038	289.880
4	1	0.647	422.825	280.877
4	2	1.000	496.385	291.528
4	3	1.000	512.331	285.820
		$n = 2000$		
1	1	0.629	828.405	567.945
1	2	1.000	1041.655	573.419
1	3	1.000	1009.698	593.316
2	1	0.634	816.994	578.041
2	2	1.000	989.527	586.232
2	3	1.000	1002.935	569.378
3	1	0.646	830.492	565.193
3	2	1.000	991.515	577.517
3	3	1.000	998.342	580.462
4	1	0.641	857.005	585.562
4	2	1.000	979.207	572.873
4	3	1.000	974.992	587.949
		$n = 3000$		
1	1	0.625	1208.155	842.016
1	2	1.000	1496.376	857.969
1	3	1.000	1462.101	884.993
2	1	0.617	1272.316	842.560
2	2	1.000	1516.441	864.525
2	3	1.000	1449.057	855.567
3	1	0.637	1251.805	860.314
3	2	1.000	1476.497	888.707
3	3	1.000	1506.054	887.445
4	1	0.620	1224.790	845.916
4	2	1.000	1439.318	852.015
4	3	1.000	1486.285	846.192

Figure 1: Base setting results with $\theta_h = 0.50$, $nbd = 1$

choiceH	choiceUpdS	SuccessRate	mean(misses)	sd(misses)
		$n = 1000$		
1	1	0.618	406.259	284.899
1	2	1.000	500.030	287.801
1	3	1.000	498.926	291.053
2	1	0.624	407.026	282.982
2	2	1.000	500.489	283.002
2	3	1.000	508.510	279.529
3	1	0.617	439.133	292.323
3	2	1.000	501.304	285.299
3	3	1.000	502.183	299.445
4	1	0.626	418.693	280.295
4	2	1.000	516.279	283.787
4	3	1.000	506.026	287.811
		$n = 2000$		
1	1	0.633	839.780	550.367
1	2	1.000	1030.876	583.853
1	3	1.000	1013.996	598.250
2	1	0.634	817.342	548.831
2	2	1.000	992.001	595.322
2	3	1.000	980.492	577.158
3	1	0.679	845.336	559.742
3	2	1.000	1032.078	572.295
3	3	1.000	974.835	595.405
4	1	0.635	804.548	537.093
4	2	1.000	986.084	579.031
4	3	1.000	1010.196	574.338
		$n = 3000$		
1	1	0.649	1242.521	858.306
1	2	1.000	1527.928	847.763
1	3	1.000	1460.141	871.720
2	1	0.626	1259.262	833.856
2	2	1.000	1485.588	862.315
2	3	1.000	1526.208	884.251
3	1	0.640	1227.862	843.167
3	2	1.000	1470.636	868.543
3	3	1.000	1524.807	866.474
4	1	0.648	1325.914	849.878
4	2	1.000	1514.407	883.120
4	3	1.000	1492.666	871.562

Figure 2: Base setting results with $\theta_h = 0.80$, $nbd = 1$

Interpretation Not much improvements as compared to the "without replacement" (choiceUpdS=2) results. We know that the expected number of misses (theoretical) for the hide-and-seek without replacement is $\frac{n-1}{2}$.

3 Experiments with different neighborhood sizes

Setting The setting for this experiment is identical to that of the base experiments in section 2. However, the neighborhood size (*nbd*) of each box is varied $\{1, 2, 3\}$. This experiment is carried out to observe if the size of neighborhood has any effect on expected number of misses when the seeker update is of type-3 (hot-cold).

Result The following table shows the effect of different *nbd* on expected number of misses. This is only applicable to update of type-3 (hot-cold).

choiceH	$nbd = 1$	$nbd = 2$	$nbd = 3$
	$n = 1000$		
1	492.347	514.478	491.224
2	507.946	505.859	490.422
3	494.038	490.952	479.129
4	512.331	488.430	504.325
	$n = 2000$		
1	1009.698	980.231	1018.029
2	1002.935	980.974	1003.716
3	998.342	976.232	982.563
4	974.992	995.226	1038.486
	$n = 3000$		
1	1462.101	1537.008	1491.913
2	1449.057	1538.320	1517.777
3	1506.054	1509.251	1532.587
4	1486.285	1537.198	1500.848

Figure 3: Effect of *nbd* on mean misses ($\theta_h = 0.50$)

choiceH	$nbd = 1$	$nbd = 2$	$nbd = 3$
	$n = 1000$		
1	498.926	485.493	486.835
2	508.510	502.662	485.133
3	502.183	495.954	488.814
4	506.026	501.933	500.365
	$n = 2000$		
1	1013.996	1048.778	981.851
2	980.492	1037.643	1059.751
3	974.835	1006.264	1017.773
4	1010.196	995.741	1001.974
	$n = 3000$		
1	1460.141	1500.349	1504.404
2	1526.208	1513.013	1546.749
3	1524.807	1518.873	1528.463
4	1492.666	1465.570	1533.059

Figure 4: Effect of *nbd* on mean misses ($\theta_h = 0.80$)

Interpretation Neighborhood does not show much effect.

4 Experiments with Hot and cold thresholds: θ_h, θ_c

Setting We thought of a setting where the seeker does not distribute its probability mass to its neighbor. This setting is more realistic, when we open a cold box. If we open a cold box then it is highly probable that the nearby boxes (its neighbors) are also cold. So, there is no point in distributing its mass to these cold neighbors. There we formalised a concept of a cold threshold (θ_c) i.e. a performance threshold where we say that if the performance of a box i is $\leq \theta_c$, then it is a cold box. We devise the following setting therefore:

1. Open a box i
2. If $perf(i) \geq \theta_h$: distribute its probability mass to all the unopened boxes in its neighbors.
3. If $perf(i) \leq \theta_c$: distribute its probability mass to all the unopened boxes except its neighbors.
4. If none of 2 or 3: distribute its probability mass to all the unopened boxes.
5. Repeat 1–4 until the hider is found.

We conduct this experiment for three different neighborhood sizes as disuseed in section 3. The thresholds are fixed at $\theta_h = 0.80$ and $\theta_c = 0.4$.

Results Here are the results:

choiceH	choiceUpdS	SuccessRate	mean(misses)	sd(misses)
		$n = 1000$		
1	1	0.647	415.765	279.076
1	2	1.000	485.743	291.479
1	3	1.000	483.394	280.817
2	1	0.622	424.259	295.290
2	2	1.000	505.753	288.841
2	3	1.000	497.388	285.391
3	1	0.614	425.721	268.688
3	2	1.000	501.925	291.147
3	3	1.000	498.737	285.369
4	1	0.629	398.860	277.083
4	2	1.000	497.892	296.853
4	3	1.000	511.561	286.341
		$n = 2000$		
1	1	0.645	854.050	568.091
1	2	1.000	1006.014	562.589
1	3	1.000	997.183	570.448
2	1	0.612	855.533	570.626
2	2	1.000	1041.809	592.339
2	3	1.000	1024.389	593.691
3	1	0.612	847.077	569.229
3	2	1.000	998.326	565.699
3	3	1.000	1016.857	574.992
4	1	0.627	820.746	542.065
4	2	1.000	1013.802	583.904
4	3	1.000	1018.995	584.717
		$n = 3000$		
1	1	0.627	1249.045	816.693
1	2	1.000	1509.410	852.052
1	3	1.000	1430.123	860.659
2	1	0.617	1255.799	812.271
2	2	1.000	1451.379	881.715
2	3	1.000	1500.294	867.022
3	1	0.617	1259.626	852.565
3	2	1.000	1513.932	873.289
3	3	1.000	1541.243	855.966
4	1	0.643	1249.299	844.649
4	2	1.000	1506.715	853.001
4	3	1.000	1501.686	880.258

Figure 5: Results for $nbd = 1$

Interpretation Similar to the θ_h update in one of the earlier section. Not much improvements as compared to the "without replacement" (choiceUpdS=2) results. We know that the expected number of misses (theoretical) for the hi de-and-seek without replacement is $\frac{n-1}{2}$.

choiceH	choiceUpdS	SuccessRate	mean(misses)	sd(misses)
		$n = 1000$		
1	1	0.616	426.451	294.068
1	2	1.000	497.421	288.863
1	3	1.000	495.362	282.265
2	1	0.647	426.195	289.582
2	2	1.000	506.020	288.400
2	3	1.000	510.959	288.912
3	1	0.597	411.358	288.290
3	2	1.000	483.829	285.909
3	3	1.000	496.718	291.624
4	1	0.647	404.587	273.281
4	2	1.000	498.959	297.005
4	3	1.000	494.474	291.465
		$n = 2000$		
1	1	0.620	832.484	558.683
1	2	1.000	1019.508	579.080
1	3	1.000	963.953	574.337
2	1	0.616	842.664	577.997
2	2	1.000	1005.651	582.930
2	3	1.000	986.946	571.343
3	1	0.630	838.575	558.411
3	2	1.000	976.245	583.901
3	3	1.000	1008.720	587.234
4	1	0.634	844.882	557.914
4	2	1.000	990.474	575.303
4	3	1.000	1048.879	587.468
		$n = 3000$		
1	1	0.607	1222.819	829.962
1	2	1.000	1461.000	864.485
1	3	1.000	1526.804	864.689
2	1	0.633	1249.983	828.909
2	2	1.000	1474.204	857.236
2	3	1.000	1520.048	851.266
3	1	0.654	1259.735	808.109
3	2	1.000	1496.244	875.656
3	3	1.000	1479.321	870.462
4	1	0.616	1223.940	866.136
4	2	1.000	1478.714	868.126
4	3	1.000	1491.112	867.572

Figure 6: Results for $nbd = 2$

choiceH	choiceUpdS	SuccessRate	mean(misses)	sd(misses)
		$n = 1000$		
1	1	0.607	419.806	284.285
1	2	1.000	506.118	288.646
1	3	1.000	500.512	286.805
2	1	0.621	427.678	290.978
2	2	1.000	480.902	297.833
2	3	1.000	492.826	282.865
3	1	0.625	401.525	277.746
3	2	1.000	510.823	288.126
3	3	1.000	500.524	289.540
4	1	0.648	432.190	285.944
4	2	1.000	499.628	287.115
4	3	1.000	500.301	294.824
		$n = 2000$		
1	1	0.620	839.482	565.586
1	2	1.000	1020.989	574.616
1	3	1.000	969.406	577.838
2	1	0.618	824.794	542.836
2	2	1.000	997.555	580.826
2	3	1.000	999.403	583.367
3	1	0.628	813.912	551.922
3	2	1.000	980.755	567.618
3	3	1.000	990.786	577.004
4	1	0.636	842.928	572.518
4	2	1.000	1007.189	573.612
4	3	1.000	989.579	584.231
		$n = 3000$		
1	1	0.636	1276.505	848.652
1	2	1.000	1485.448	870.575
1	3	1.000	1459.934	855.671
2	1	0.619	1200.735	843.910
2	2	1.000	1471.909	877.663
2	3	1.000	1572.459	862.723
3	1	0.623	1295.474	884.821
3	2	1.000	1473.981	869.361
3	3	1.000	1481.696	887.774
4	1	0.630	1239.613	840.033
4	2	1.000	1515.664	876.244
4	3	1.000	1467.864	864.726

Figure 7: Results for $nbd = 3$