Results of the AI1 Kalah Tournament

ABSTRACT

This report contains the results of the closed AI1 Kalah tournament. All teams that manage to pass the first stage will receive bonus points. The top ten teams receive additional bonus points. The tournament consists of multiple stages, where agents are disqualified if they don't perform well enough.

1. Stage "Sanity Test"

All agents are made to compete once against a random bot on a (6, 6) board. As the agent is allowed to make the first move, we know that they must be able to win the game, since this configuration of Kalah is solved. To pass this stage, one has to win a best out of three against the random bot, otherwise one is disqualified immediately.

1.1. Scores

Agent	Win	Loss	Draw	Score
mo57fery	**1**	**2**	**0**	**2**
MCTS but in bad	**3**	**0**	**0**	**6**
Nameless	**3**	**0**	**0**	**6**
AiAiAiAi	**3**	**0**	**0**	**6**
Carla, ey, Ei!	**3**	**0**	**0**	**6**
MyAgent	**3**	**0**	**0**	**6**
Stirling Archer	**3**	**0**	**0**	**6**
yahiko	**3**	**0**	**0**	**6**
ai-1	**3**	**0**	**0**	**6**
ku81tuli;uj59ames;ez03odak	**3**	**0**	**0**	**6**
TheBestAgent	**3**	**0**	**0**	**6**
Agent of Chaos	**3**	**0**	**0**	**6**
Stickfish	**3**	**0**	**0**	**6**
na05kymu;ce84pegu	**3**	**0**	**0**	**6**
jo43beqy	**3**	**0**	**0**	**6**
EdizDerBreite	**3**	**0**	**0**	**6**

1.2. Game Log

Nr.	South Agent	North Agent	South	North	Diff.
1	**yahiko**	**Random**	**57**	**15**	**42**
2	**Nameless**	**Random**	**52**	**20**	**32**
3	**TheBestAgent**	**Random**	**48**	**24**	**24**
4	**MyAgent**	**Random**	**44**	**28**	**16**
5	**TheBestAgent**	**Random**	**58**	**14**	**44**
6	**na05kymu;ce84pegu**	**Random**	**47**	**25**	**22**
7	**EdizDerBreite**	**Random**	**60**	**12**	**48**
8	**ku81tuli;uj59ames;ez03odak**	**Random**	**43**	**29**	**14**
9	**Stirling Archer**	**Random**	**48**	**24**	**24**

10	**jo43beqy**	**Random**	**43**	**29**	**14**
11	**Carla, ey, Ei!**	**Random**	**53**	**19**	**34**
12	**Nameless**	**Random**	**57**	**15**	**42**
13	**MyAgent**	**Random**	**59**	**13**	**46**
14	**Stirling Archer**	**Random**	**53**	**19**	**34**
15	**Nameless**	**Random**	**57**	**15**	**42**
16	**Stirling Archer**	**Random**	**49**	**23**	**26**
17	**Stickfish**	**Random**	**52**	**20**	**32**
18	**Agent of Chaos**	**Random**	**43**	**29**	**14**
19	**yahiko**	**Random**	**58**	**14**	**44**
20	**EdizDerBreite**	**Random**	**48**	**24**	**24**
21	**MCTS but in bad**	**Random**	**55**	**17**	**38**
22	**na05kymu;ce84pegu**	**Random**	**51**	**21**	**30**
23	**jo43beqy**	**Random**	**44**	**28**	**16**
24	**ai-1**	**Random**	**49**	**23**	**26**
25	**jo43beqy**	**Random**	**42**	**30**	**12**
26	**Carla, ey, Ei!**	**Random**	**59**	**13**	**46**
27	**ku81tuli;uj59ames;ez03odak**	**Random**	**62**	**10**	**52**
28	**mo57fery**	**Random**	**38**	**34**	**4**
29	**mo57fery**	**Random**	**34**	**38**	**-4**
30	**Carla, ey, Ei!**	**Random**	**48**	**24**	**24**
31	**na05kymu;ce84pegu**	**Random**	**49**	**23**	**26**
32	**TheBestAgent**	**Random**	**57**	**15**	**42**
33	**Agent of Chaos**	**Random**	**53**	**19**	**34**
34	**MCTS but in bad**	**Random**	**54**	**18**	**36**
35	**mo57fery**	**Random**	**30**	**42**	**-12**
36	**MyAgent**	**Random**	**45**	**27**	**18**
37	**ku81tuli;uj59ames;ez03odak**	**Random**	**60**	**12**	**48**
38	**yahiko**	**Random**	**43**	**29**	**14**
39	**Agent of Chaos**	**Random**	**50**	**22**	**28**
40	**AiAiAiAi**	**Random**	**56**	**16**	**40**
41	**Stickfish**	**Random**	**61**	**11**	**50**
42	**EdizDerBreite**	**Random**	**59**	**13**	**46**
43	**AiAiAiAi**	**Random**	**61**	**11**	**50**
44	**AiAiAiAi**	**Random**	**43**	**29**	**14**
45	**MCTS but in bad**	**Random**	**46**	**26**	**20**
46	**Stickfish**	**Random**	**50**	**22**	**28**
47	**ai-1**	**Random**	**47**	**25**	**22**
48	**ai-1**	**Random**	**41**	**31**	**10**

These agents were disqualified for failing to meet the necessary criteria for proceeding to the next round:

- mo57fery

2. Stage "Round Robin (6, 6)"

All agents play against all other agents, on both sides of a Kalah board. If one agent definitively manages to beat another agent, they are awarded two points, and the opponent is given no points. For a draw, both agents are granted a single point. The final score of this round is calculated by summing up the points for each game. Agents that suffered more losses than wins are disqualifed.

2.1. Scores

Agent	Win	Loss	Draw	Score
jo43beqy	0	28	0	0
Agent of Chaos	5	23	0	10
ai-1	5	21	2	12
yahiko	7	19	1	15
TheBestAgent	7	20	1	15
AiAiAiAi	8	20	0	16
Nameless	12	14	0	24
MyAgent	12	13	1	25
na05kymu;ce84pegu	14	12	0	28
ku81tuli;uj59ames;ez03odak	14	12	1	29
Stirling Archer	16	11	0	32
EdizDerBreite	23	4	0	46
MCTS but in bad	23	4	1	47
Stickfish	23	2	1	47
Carla, ey, Ei!	25	3	0	50

2.2. Game Log

Nr.	South Agent	North Agent	South	North	Diff.
1	jo43beqy	MyAgent	23	49	-26
2	MCTS but in bad	Agent of Chaos	51	21	30
3	Agent of Chaos	yahiko	29	43	-14
4	yahiko	na05kymu;ce84pegu	29	43	-14
5	Nameless	ai-1	54	18	36
6	Agent of Chaos	EdizDerBreite	20	52	-32
7	MyAgent	yahiko	33	39	-6
8	Agent of Chaos	AiAiAiAi	42	30	12
9	MCTS but in bad	EdizDerBreite	41	31	10
10	TheBestAgent	Stickfish	27	45	-18
11	MCTS but in bad	Stickfish	24	48	-24
12	MCTS but in bad	na05kymu;ce84pegu	42	30	12
13	Carla, ey, Ei!	ai-1	54	18	36
14	Stirling Archer	MyAgent	43	29	14
15	na05kymu;ce84pegu	AiAiAiAi	44	28	16
16	MyAgent	Stirling Archer	21	34	-13
17	ai-1	MCTS but in bad	17	55	-38
18	ku81tuli;uj59ames;ez03odak	AiAiAiAi	33	39	-6
19	Stickfish	MyAgent	41	31	10
20	EdizDerBreite	ku81tuli;uj59ames;ez03odak	46	26	20
21	TheBestAgent	Agent of Chaos	43	29	14
22	ai-1	Stickfish	33	39	-6
23	Stickfish	jo43beqy	49	23	26
24	ku81tuli;uj59ames;ez03odak	MCTS but in bad	29	43	-14
25	MyAgent	na05kymu;ce84pegu	17	35	-18
26	Stickfish	Stirling Archer	41	31	10
27	na05kymu;ce84pegu	MCTS but in bad	31	41	-10
28	jo43beqy	na05kymu;ce84pegu	18	54	-36
29	MCTS but in bad	MyAgent	37	35	2
30	Agent of Chaos	Stickfish	27	45	-18
31	jo43beqy	TheBestAgent	24	48	-24
32	yahiko	jo43beqy	52	20	32
33	Nameless	jo43beqy	50	22	28

34	**ai-1**	**Stirling Archer**	**19**	**53**	**-34**
35	**yahiko**	**Carla, ey, Ei!**	**18**	**54**	**-36**
36	**yahiko**	**MyAgent**	**20**	**52**	**-32**
37	**jo43beqy**	**Nameless**	**22**	**50**	**-28**
38	**na05kymu;ce84pegu**	**Carla, ey, Ei!**	**32**	**40**	**-8**
39	**EdizDerBreite**	**yahiko**	**58**	**14**	**44**
40	**Agent of Chaos**	**TheBestAgent**	**34**	**38**	**-4**
41	**na05kymu;ce84pegu**	**TheBestAgent**	**37**	**35**	**2**
42	**Agent of Chaos**	**Nameless**	**17**	**55**	**-38**
43	**Stickfish**	**Nameless**	**47**	**25**	**22**
44	**na05kymu;ce84pegu**	**MyAgent**	**42**	**30**	**12**
45	**Nameless**	**AiAiAiAi**	**49**	**23**	**26**
46	**jo43beqy**	**Agent of Chaos**	**32**	**40**	**-8**
47	**TheBestAgent**	**Carla, ey, Ei!**	**21**	**51**	**-30**
48	**Nameless**	**Stirling Archer**	**19**	**53**	**-34**
49	**na05kymu;ce84pegu**	**Stickfish**	**34**	**38**	**-4**
50	**ku81tuli;uj59ames;ez03odak**	**TheBestAgent**	**47**	**25**	**22**
51	**Agent of Chaos**	**Stirling Archer**	**22**	**50**	**-28**
52	**na05kymu;ce84pegu**	**Agent of Chaos**	**45**	**27**	**18**
53	**ai-1**	**na05kymu;ce84pegu**	**24**	**48**	**-24**
54	**yahiko**	**Agent of Chaos**	**43**	**29**	**14**
55	**MCTS but in bad**	**jo43beqy**	**52**	**20**	**32**
56	**ku81tuli;uj59ames;ez03odak**	**Stickfish**	**28**	**44**	**-16**
57	**ai-1**	**MyAgent**	**14**	**58**	**-44**
58	**yahiko**	**MCTS but in bad**	**20**	**52**	**-32**
59	**Stickfish**	**na05kymu;ce84pegu**	**45**	**27**	**18**
60	**Agent of Chaos**	**ku81tuli;uj59ames;ez03odak**	**30**	**42**	**-12**
61	**MCTS but in bad**	**ai-1**	**46**	**26**	**20**
62	**ku81tuli;uj59ames;ez03odak**	**Carla, ey, Ei!**	**34**	**38**	**-4**
63	**AiAiAiAi**	**Nameless**	**19**	**53**	**-34**
64	**EdizDerBreite**	**Stickfish**	**34**	**23**	**11**
65	**MCTS but in bad**	**Carla, ey, Ei!**	**33**	**39**	**-6**
66	**ku81tuli;uj59ames;ez03odak**	**jo43beqy**	**45**	**27**	**18**
67	**yahiko**	**ai-1**	**31**	**41**	**-10**
68	**Carla, ey, Ei!**	**jo43beqy**	**58**	**14**	**44**
69	**MyAgent**	**ku81tuli;uj59ames;ez03odak**	**36**	**36**	**0**
70	**ku81tuli;uj59ames;ez03odak**	**Stirling Archer**	**19**	**35**	**-16**
71	**TheBestAgent**	**MCTS but in bad**	**20**	**52**	**-32**
72	**TheBestAgent**	**ai-1**	**19**	**53**	**-34**
73	**EdizDerBreite**	**MCTS but in bad**	**49**	**23**	**26**
74	**jo43beqy**	**Carla, ey, Ei!**	**24**	**48**	**-24**
75	**Agent of Chaos**	**MCTS but in bad**	**23**	**49**	**-26**
76	**Agent of Chaos**	**Carla, ey, Ei!**	**21**	**51**	**-30**
77	**ai-1**	**yahiko**	**36**	**36**	**0**
78	**AiAiAiAi**	**Carla, ey, Ei!**	**29**	**43**	**-14**
79	**jo43beqy**	**Stirling Archer**	**10**	**62**	**-52**
80	**ku81tuli;uj59ames;ez03odak**	**MyAgent**	**37**	**35**	**2**
81	**jo43beqy**	**EdizDerBreite**	**13**	**59**	**-46**
82	**jo43beqy**	**yahiko**	**21**	**51**	**-30**
83	**MyAgent**	**jo43beqy**	**46**	**26**	**20**
84	**yahiko**	**EdizDerBreite**	**20**	**52**	**-32**
85	**MyAgent**	**Agent of Chaos**	**54**	**18**	**36**
86	**AiAiAiAi**	**EdizDerBreite**	**27**	**45**	**-18**

87	ku81tuli;uj59ames;ez03odak	na05kymu;ce84pegu	45	27	18
88	EdizDerBreite	Agent of Chaos	48	24	24
89	Stickfish	yahiko	43	29	14
90	Carla, ey, Ei!	TheBestAgent	44	28	16
91	Stickfish	MCTS but in bad	36	36	0
92	EdizDerBreite	Nameless	47	25	22
93	na05kymu;ce84pegu	Stirling Archer	27	31	-4
94	Carla, ey, Ei!	AiAiAiAi	40	32	8
95	jo43beqy	Stickfish	32	40	-8
96	MyAgent	Nameless	24	48	-24
97	Carla, ey, Ei!	na05kymu;ce84pegu	44	28	16
98	MCTS but in bad	AiAiAiAi	47	25	22
99	Stirling Archer	AiAiAiAi	45	27	18
100	Stickfish	ai-1	55	17	38
101	ai-1	Nameless	21	51	-30
102	TheBestAgent	MyAgent	28	44	-16
103	AiAiAiAi	ku81tuli;uj59ames;ez03odak	31	41	-10
104	ku81tuli;uj59ames;ez03odak	ai-1	51	21	30
105	na05kymu;ce84pegu	jo43beqy	46	26	20
106	Nameless	Carla, ey, Ei!	32	40	-8
107	TheBestAgent	Nameless	38	34	4
108	jo43beqy	AiAiAiAi	30	42	-12
109	Nameless	MCTS but in bad	27	45	-18
110	Agent of Chaos	MyAgent	27	45	-18
111	Nameless	Stickfish	27	45	-18
112	ai-1	AiAiAiAi	29	43	-14
113	AiAiAiAi	Stickfish	30	42	-12
114	Stickfish	Carla, ey, Ei!	35	37	-2
115	yahiko	Stickfish	19	53	-34
116	MyAgent	Carla, ey, Ei!	28	44	-16
117	Stirling Archer	Carla, ey, Ei!	31	41	-10
118	Carla, ey, Ei!	Stirling Archer	42	30	12
119	MyAgent	AiAiAiAi	48	24	24
120	AiAiAiAi	Agent of Chaos	57	15	42
121	MyAgent	TheBestAgent	51	21	30
122	EdizDerBreite	ai-1	51	21	30
123	ai-1	TheBestAgent	36	36	0
124	Nameless	TheBestAgent	34	23	11
125	Stickfish	EdizDerBreite	24	48	-24
126	yahiko	AiAiAiAi	32	40	-8
127	yahiko	Stirling Archer	20	52	-32
128	Stirling Archer	EdizDerBreite	26	30	-4
129	EdizDerBreite	AiAiAiAi	40	32	8
130	Stirling Archer	TheBestAgent	45	27	18
131	ai-1	EdizDerBreite	11	61	-50
132	AiAiAiAi	jo43beqy	50	22	28
133	Nameless	ku81tuli;uj59ames;ez03odak	37	35	2
134	TheBestAgent	AiAiAiAi	37	35	2
135	Stickfish	TheBestAgent	39	33	6
136	Stirling Archer	yahiko	44	28	16
137	ai-1	Carla, ey, Ei!	22	50	-28
138	MyAgent	ai-1	45	27	18
139	MCTS but in bad	yahiko	51	21	30

140	**na05kymu;ce84pegu**	**Nameless**	**42**	**30**	**12**
141	**Nameless**	**MyAgent**	**34**	**38**	**-4**
142	**Agent of Chaos**	**jo43beqy**	**50**	**22**	**28**
143	**Carla, ey, Ei!**	**yahiko**	**50**	**22**	**28**
144	**Stirling Archer**	**ai-1**	**44**	**28**	**16**
145	**ku81tuli;uj59ames;ez03odak**	**Nameless**	**42**	**30**	**12**
146	**Nameless**	**Agent of Chaos**	**44**	**28**	**16**
147	**jo43beqy**	**MCTS but in bad**	**19**	**53**	**-34**
148	**ku81tuli;uj59ames;ez03odak**	**EdizDerBreite**	**26**	**46**	**-20**
149	**yahiko**	**Nameless**	**15**	**57**	**-42**
150	**Stickfish**	**Agent of Chaos**	**44**	**28**	**16**
151	**na05kymu;ce84pegu**	**ku81tuli;uj59ames;ez03odak**	**33**	**39**	**-6**
152	**TheBestAgent**	**EdizDerBreite**	**31**	**41**	**-10**
153	**Nameless**	**yahiko**	**48**	**24**	**24**
154	**Carla, ey, Ei!**	**ku81tuli;uj59ames;ez03odak**	**46**	**26**	**20**
155	**Nameless**	**EdizDerBreite**	**27**	**28**	**-1**
156	**MCTS but in bad**	**Stirling Archer**	**37**	**35**	**2**
157	**MyAgent**	**Stickfish**	**31**	**41**	**-10**
158	**ai-1**	**ku81tuli;uj59ames;ez03odak**	**35**	**37**	**-2**
159	**Stirling Archer**	**jo43beqy**	**41**	**31**	**10**
160	**TheBestAgent**	**ku81tuli;uj59ames;ez03odak**	**28**	**44**	**-16**
161	**MCTS but in bad**	**Nameless**	**41**	**31**	**10**
162	**MCTS but in bad**	**TheBestAgent**	**47**	**25**	**22**
163	**yahiko**	**TheBestAgent**	**39**	**33**	**6**
164	**Stirling Archer**	**ku81tuli;uj59ames;ez03odak**	**42**	**30**	**12**
165	**MCTS but in bad**	**ku81tuli;uj59ames;ez03odak**	**42**	**30**	**12**
166	**TheBestAgent**	**na05kymu;ce84pegu**	**29**	**43**	**-14**
167	**EdizDerBreite**	**TheBestAgent**	**40**	**32**	**8**
168	**Agent of Chaos**	**na05kymu;ce84pegu**	**30**	**42**	**-12**
169	**AiAiAiAi**	**yahiko**	**58**	**14**	**44**
170	**jo43beqy**	**ku81tuli;uj59ames;ez03odak**	**22**	**50**	**-28**
171	**Stirling Archer**	**Stickfish**	**36**	**22**	**14**
172	**Stirling Archer**	**Agent of Chaos**	**40**	**32**	**8**
173	**AiAiAiAi**	**MCTS but in bad**	**27**	**45**	**-18**
174	**TheBestAgent**	**jo43beqy**	**39**	**33**	**6**
175	**Carla, ey, Ei!**	**Agent of Chaos**	**54**	**18**	**36**
176	**MyAgent**	**EdizDerBreite**	**31**	**41**	**-10**
177	**na05kymu;ce84pegu**	**yahiko**	**47**	**25**	**22**
178	**Stirling Archer**	**MCTS but in bad**	**29**	**43**	**-14**
179	**AiAiAiAi**	**TheBestAgent**	**42**	**30**	**12**
180	**MyAgent**	**MCTS but in bad**	**35**	**37**	**-2**
181	**Carla, ey, Ei!**	**EdizDerBreite**	**35**	**37**	**-2**
182	**Carla, ey, Ei!**	**MCTS but in bad**	**45**	**27**	**18**
183	**na05kymu;ce84pegu**	**EdizDerBreite**	**21**	**51**	**-30**
184	**TheBestAgent**	**Stirling Archer**	**40**	**32**	**8**
185	**EdizDerBreite**	**Carla, ey, Ei!**	**42**	**30**	**12**
186	**yahiko**	**ku81tuli;uj59ames;ez03odak**	**40**	**32**	**8**
187	**AiAiAiAi**	**na05kymu;ce84pegu**	**31**	**25**	**6**
188	**Stickfish**	**ku81tuli;uj59ames;ez03odak**	**41**	**31**	**10**
189	**AiAiAiAi**	**MyAgent**	**13**	**59**	**-46**
190	**Stirling Archer**	**na05kymu;ce84pegu**	**40**	**32**	**8**
191	**AiAiAiAi**	**Stirling Archer**	**26**	**46**	**-20**
192	**EdizDerBreite**	**MyAgent**	**16**	**13**	**3**

193	TheBestAgent	yahiko	32	29	3
194	jo43beqy	ai-1	27	45	-18
195	AiAiAiAi	ai-1	23	49	-26
196	Agent of Chaos	ai-1	38	34	4
197	ai-1	Agent of Chaos	33	39	-6
198	Carla, ey, Ei!	MyAgent	40	32	8
199	EdizDerBreite	na05kymu;ce84pegu	45	27	18
200	EdizDerBreite	jo43beqy	52	20	32
201	na05kymu;ce84pegu	ai-1	49	23	26
202	ku81tuli;uj59ames;ez03odak	Agent of Chaos	43	29	14
203	Nameless	na05kymu;ce84pegu	33	39	-6
204	EdizDerBreite	Stirling Archer	41	31	10
205	ku81tuli;uj59ames;ez03odak	yahiko	50	22	28
206	Carla, ey, Ei!	Nameless	47	25	22
207	Stickfish	AiAiAiAi	48	24	24
208	ai-1	jo43beqy	43	29	14
209	Stirling Archer	Nameless	50	22	28
210	Carla, ey, Ei!	Stickfish	33	39	-6

These agents were disqualified for failing to meet the necessary criteria for proceeding to the next round:

- Nameless
- AiAiAiAi
- yahiko
- ai-1
- TheBestAgent
- Agent of Chaos
- jo43beqy

3. Stage "Round Robin (8, 8)"

All agents play against all other agents, on both sides of a Kalah board. If one agent definitively manages to beat another agent, they are awarded two points, and the opponent is given no points. For a draw, both agents are granted a single point. The final score of this round is calculated by summing up the points for each game. Agents that suffered more losses than wins are disqualifed.

3.1. Scores

Agent	Win	Loss	Draw	Score
MyAgent	1	12	0	2
ku81tuli;uj59ames;ez03odak	1	11	1	3
na05kymu;ce84pegu	4	9	0	8
MCTS but in bad	6	8	0	12
Carla, ey, Ei!	7	5	1	15
Stirling Archer	8	6	0	16
Stickfish	11	3	0	22
EdizDerBreite	13	1	0	26

3.2. Game Log

Nr.	South Agent	North Agent	South	North	Diff.
1	MyAgent	EdizDerBreite	42	86	-44

2	Stickfish	Stirling Archer	54	74	-20
3	na05kymu;ce84pegu	MyAgent	76	52	24
4	Stirling Archer	ku81tuli;uj59ames;ez03odak	73	55	18
5	Carla, ey, Ei!	EdizDerBreite	51	77	-26
6	EdizDerBreite	MCTS but in bad	90	38	52
7	na05kymu;ce84pegu	ku81tuli;uj59ames;ez03odak	80	48	32
8	EdizDerBreite	Carla, ey, Ei!	68	60	8
9	Stickfish	EdizDerBreite	74	54	20
10	MyAgent	na05kymu;ce84pegu	63	65	-2
11	na05kymu;ce84pegu	EdizDerBreite	60	68	-8
12	MyAgent	Stirling Archer	57	60	-3
13	MCTS but in bad	MyAgent	76	52	24
14	Carla, ey, Ei!	na05kymu;ce84pegu	78	50	28
15	Stirling Archer	EdizDerBreite	55	73	-18
16	Stickfish	Carla, ey, Ei!	74	54	20
17	Stickfish	MyAgent	72	56	16
18	Stickfish	ku81tuli;uj59ames;ez03odak	84	44	40
19	ku81tuli;uj59ames;ez03odak	MyAgent	73	55	18
20	ku81tuli;uj59ames;ez03odak	na05kymu;ce84pegu	59	63	-4
21	ku81tuli;uj59ames;ez03odak	EdizDerBreite	26	102	-76
22	Stirling Archer	Carla, ey, Ei!	73	55	18
23	ku81tuli;uj59ames;ez03odak	MCTS but in bad	62	66	-4
24	MCTS but in bad	Stirling Archer	51	77	-26
25	MCTS but in bad	EdizDerBreite	56	72	-16
26	na05kymu;ce84pegu	MCTS but in bad	60	68	-8
27	Stickfish	na05kymu;ce84pegu	89	39	50
28	MyAgent	MCTS but in bad	51	77	-26
29	Stirling Archer	na05kymu;ce84pegu	64	50	14
30	MCTS but in bad	Stickfish	55	73	-18
31	ku81tuli;uj59ames;ez03odak	Carla, ey, Ei!	64	64	0
32	na05kymu;ce84pegu	Stirling Archer	68	60	8
33	Carla, ey, Ei!	Stirling Archer	90	38	52
34	MyAgent	Carla, ey, Ei!	55	73	-18
35	na05kymu;ce84pegu	Stickfish	52	76	-24
36	Carla, ey, Ei!	MyAgent	76	52	24
37	MCTS but in bad	ku81tuli;uj59ames;ez03odak	76	52	24
38	Stirling Archer	MCTS but in bad	73	55	18
39	MyAgent	Stickfish	60	68	-8
40	EdizDerBreite	Stirling Archer	86	42	44
41	MCTS but in bad	na05kymu;ce84pegu	74	54	20
42	na05kymu;ce84pegu	Carla, ey, Ei!	63	48	15
43	Carla, ey, Ei!	Stickfish	60	68	-8
44	EdizDerBreite	MyAgent	90	38	52
45	MyAgent	ku81tuli;uj59ames;ez03odak	84	44	40
46	Carla, ey, Ei!	MCTS but in bad	80	48	32
47	Stirling Archer	Stickfish	68	60	8
48	EdizDerBreite	na05kymu;ce84pegu	90	38	52
49	EdizDerBreite	Stickfish	73	55	18
50	EdizDerBreite	ku81tuli;uj59ames;ez03odak	82	46	36
51	Stickfish	MCTS but in bad	80	48	32
52	MCTS but in bad	Carla, ey, Ei!	56	72	-16
53	ku81tuli;uj59ames;ez03odak	Stickfish	53	75	-22
54	Stirling Archer	MyAgent	76	52	24

55	**Carla, ey, Ei!**	**ku81tuli;uj59ames;ez03odak**	**74**	**54**	**20**
56	**ku81tuli;uj59ames;ez03odak**	**Stirling Archer**	**61**	**67**	**-6**

These agents were disqualified for failing to meet the necessary criteria for proceeding to the next round:

- MyAgent
- na05kymu;ce84pegu
- ku81tuli;uj59ames;ez03odak
- MCTS but in bad

4. Stage "Round Robin (10, 10)"

All agents play against all other agents, on both sides of a Kalah board. If one agent definitively manages to beat another agent, they are awarded two points, and the opponent is given no points. For a draw, both agents are granted a single point. The final score of this round is calculated by summing up the points for each game. Agents that suffered more losses than wins are disqualifed.

4.1. Scores

Agent	Win	Loss	Draw	Score
Carla, ey, Ei!	**1**	**5**	**0**	**2**
Stirling Archer	**1**	**5**	**0**	**2**
Stickfish	**5**	**1**	**0**	**10**
EdizDerBreite	**5**	**1**	**0**	**10**

4.2. Game Log

Nr.	South Agent	North Agent	South	North	Diff.
1	**Carla, ey, Ei!**	**Stickfish**	**80**	**120**	**-40**
2	**Stirling Archer**	**EdizDerBreite**	**87**	**113**	**-26**
3	**Stickfish**	**Carla, ey, Ei!**	**107**	**93**	**14**
4	**EdizDerBreite**	**Stirling Archer**	**117**	**83**	**34**
5	**EdizDerBreite**	**Carla, ey, Ei!**	**123**	**77**	**46**
6	**Stickfish**	**EdizDerBreite**	**89**	**111**	**-22**
7	**Carla, ey, Ei!**	**EdizDerBreite**	**92**	**108**	**-16**
8	**Carla, ey, Ei!**	**Stirling Archer**	**115**	**85**	**30**
9	**Stirling Archer**	**Carla, ey, Ei!**	**121**	**79**	**42**
10	**Stirling Archer**	**Stickfish**	**87**	**113**	**-26**
11	**Stickfish**	**Stirling Archer**	**116**	**84**	**32**
12	**EdizDerBreite**	**Stickfish**	**96**	**104**	**-8**

These agents were disqualified for failing to meet the necessary criteria for proceeding to the next round:

- Carla, ey, Ei!
- Stirling Archer

5. Stage "Round Robin (12, 12)"

All agents play against all other agents, on both sides of a Kalah board. If one agent definitively manages to beat another agent, they are awarded two points, and the opponent is given no points. For a draw, both agents are granted a single point. The final score of this round is calculated by summing up the points for each game. Agents that suffered more losses than wins are disqualifed.

5.1. Scores

Agent	Win	Loss	Draw	Score
Stickfish	**0**	**2**	**0**	**0**
EdizDerBreite	**2**	**0**	**0**	**4**

5.2. Game Log

Nr.	South Agent	North Agent	South	North	Diff.
1	**Stickfish**	**EdizDerBreite**	**135**	**153**	**-18**
2	**EdizDerBreite**	**Stickfish**	**149**	**139**	**10**

These agents were disqualified for failing to meet the necessary criteria for proceeding to the next round:

- Stickfish

6. Final score

The top ten agents are as follows:

1 EdizDerBreite (Score: 92)

2 Stickfish (Score: 85)

3 Carla, ey, Ei! (Score: 73)

4 MCTS but in bad (Score: 65)

5 Stirling Archer (Score: 56)

6 na05kymu;ce84pegu (Score: 42)

7 ku81tuli;uj59ames;ez03odak (Score: 38)

8 MyAgent (Score: 33)

9 Nameless (Score: 30)

10 AiAiAiAi (Score: 22)

Congratulations to all participating teams!

The remaining scores are:

11 yahiko (Score: 21)

11 TheBestAgent (Score: 21)

12 ai-1 (Score: 18)

13 Agent of Chaos (Score: 16)

14 jo43beqy (Score: 6)

15 mo57fery (Score: 2)

7. About the agents

This section contains the abridged, partly spellchecked and expanded (where *needed*) contents of the ABOUT file, if it was submitted. The agent names are taken from the submitted code where it was possible. Else the name consists of the idm ids of the submitters.

7.1. mo57fery (224972)

Make the game a minmax problem and choose the best value to continue to play, without any assumption that the other makes a mistake.

The minmax function calls itself recursively. It searches the best moves and the best value.

7.2. MCTS but in bad (223785)

Idea

Monte Carlo tree search

Implementation

Concepts used:

- MCTS with board heuristic instead of playouts
- UCT-Score for search guidance (c = 2.5)
- Min-Max Backup

    The ideas came from the paper ‘‘Trade-Offs in Sampling-Based Adversarial Planning‘‘.

    Saving the built tree to avoid recomputation was attempted, but the overhead created by the multiprocessing manager made in inpractical.

7.3. Nameless (224092)

Idea

The idea is to use MCTS to search the game tree via biased sampling and then take the move, which seems to lead to best mean outcomes.

Implementation

Our agent was written as follows: A combination of Monte-Carlo-Tree-Search and Min-Max with Alpha-Beta-Pruning was used. The agent starts with mcts and switches at a specific state of the game to min-max. The turning point is chosen by the number of seeds left in the game. Mcts should perform better at the beginning of the game since the branching factor is probably too high for min-max to give good results. Mcts develops a good intuition of the game and min-max finishes the game better.

The tree for mcts is build out of Node objects, which save the board state, the q value, the number of visits, their parent node, their explored children and the unexplored legal moves. With two convenience functions, one to add a rollout result and one to extend the tree by a node. The remaining agent consists of an uct_score function to calculate the score used to select the best child, a select function which uses the uct score to select the best child, a rollout function, which follows a random policy until the game terminates, a backprop_result function which backpropagates the result of the rollout up the tree, an evaluate function, which calculate the differences in stores between both players and a q function, which decides base on this difference, whether its a win, a tie or a loss. The one level above is the mcts_search function, which either extends a node, starts a rollout and then backpropagates the result or selects the best child, if all the nodes children are already extended. This is done in a time restricted loop and once the time is up the action with the best mean outcome will be returned. At the highest level the agent generator (which yields the mcts_search) is called with the actual board state as `root_node`. This can be easily improved by taking the child of the previous root corresponding to the done action. This enables to reuse some of the previous search effort, but also takes up more RAM.

The min-max search given was adapted with an evaluation which counts the number of seeds a player has captured minus the number of seeds an opponent has captured and also takes into account the difference of seeds still in the pits between the opponent and us. Furthermore alpha-beta pruning was added to the min-max algorithm with variable ordering by evaluating the next board state.

7.4. AiAiAiAi (224180)

Idea

Basic min/max algorithm with alpha/beta pruning but without more sophisticated methods.

Implementation

Simple evaluation function that takes into account the tokens in the stores with a factor for scaling, simple minimax algorithm like presented in the lecture. The agent was written in Python.

7.5. Carla, ey, Ei! (224221)

Idea

We started in Python, then switched to Golang because of the lack of speed in Python. First we implemented a simple MinMax algorithm with alpha-beta pruning. To deliver multiple outputs in the given time, we included iterative deepening. As we read more about the game theory of Kalah, we turned the MinMax algorithm into a NegaMax algorithm, because Kalah is a zero-sum game. After further reading we implemented a simple move order to speed up the search by using more effective pruning. In Python we also tested an implementation of a transposition table that stores game states and their best moves. This was difficult because kgp.py used processes rather than threads, so the table was not saved between searches. So we switched to principal variation search, which assumes that the first move in the order is the best, and is even more effective at pruning. But in the worst case it is as fast as NegaMax. This is where Python reached its limits and we switched to a language that is compiled. Go was a good option because it had a really good tutorial and an already implemented Kalah game protocol.

Implementation

We do iterative deepening on the `Search` function There we use the principal variation search based on the pseudocode on wikipedia with the following parameters:

- `state` (the game state of the node)
- `side` (the player whose turn it is)
- `depth` (the search depth)
- `alpha` (value for pruning)
- `beta` (value for pruning)

For the evaluation we chose a rather simple heuristic which is just the score difference.

Now comes the main part of the principal variation search: Inside the for loop we traverse all possible moves for the given state. Before searching further nodes we check if the enemy has already won, and if so we continue with the next move. First we evaluate the first move we got from ordering. Here we search the child nodes recursively. If the move grants us another turn, we leave the alpha and beta bounds as they are. Otherwise we search as in NegaMax with alpha = -beta and beta = -alpha. The score is also negated. Next we search all other moves in the order, but with a smaller search window for alpha = -alpha-1 and beta = -alpha. If the search fails high, meaning that the calculated score is between the initial alpha and beta, we have to search the nodes with the boundaries again as we searched the first move.

Before searching the next move in the for loop we compare the alpha and the calculated score. If the new score is better, we update alpha and bestMove. And here we prune if the alpha is greater than or equal to the beta.

Finally we return `alpha` and the `bestMove`.

7.6. MyAgent (224225)

Idea

Minimax-search with iterative deepening search

Implementation

We implemented a negamax algorithm which is basically the minimax-algorithm where min and max use the same function just with different signs. We also tried to implement alpha-beta pruning but couldn't get that working. Thus we are submitting the agent without pruning. We programmed the agent in java using recursion.

7.7. Stirling Archer (224474)

Idea

The agent performs a iterative deepening search applying multiple performance enhancements. First off alpha beta pruning is used to reduce the expanded part of the search tree. To further improve the effectiveness of the pruning move ordering is implemented. We try to predict the potential for each allowed move by looking at move properties that can be directly observed: Properties seen as good are (in descending order):

1. the player is allowed to take another move directly after this move (this is essentially a ‘‘free‘‘ move)
2. the move captures a pit (thus ‘‘steeling‘‘ seeds from the enemy side and securing multiple seeds in one move)
3. the move added seeds to the players store (going one or multiple times over the own store)

Moves are expanded in this order. Moves that do not have any of these properties are expanded last.

Our evaluation function takes three key measures into account:

First: the difference between the own store and the enemy store. Obviously this is a clear indicator if one side is closer to winning.

Second: the number of seeds on the enemy side. Seeds on the enemy side are potentially easier to get into the enemy store. This is not as strong an indicator as the seeds that are already in the stores an thus weighted lower.

Third: if one player already won, all other metrics should be irrelevant and a huge constant is added. This ranks winning moves always above non-winning ones but also allows to distinguish between ‘‘better‘‘ and ‘‘worse‘‘ wins. This is relevant in the case that it an enemy move was made on this branch. We assumed the enemy to take the best move possible according to **our** evaluation function. Since the enemy might have a different evaluation function it might take a different path than we assumed that might have truncated. Using a combination of the non-winning-evaluation and the winning constant allows us the estimate to overall-goodness of the direction of a (potentially winning) move.

Implementation

Our agent is implemented in python and uses the provided template. For the iterative deepening we start with depth one. This depth should be reached under all hardware conditions and bord sizes. After that we continue with depth two and increase the depth after each successful run by two.

The calculation of the best move for each depth is done in `search_alpha_beta` which calls itself recursively. For each recursive call we take into account, whether the current player is allowed to make another move or not. If the player is is allowed to make another move we calculate the next best move by calling `search_alpha_beta` again with the same parameter (but progressed bord state). If the player is not allowed to make another move `search_alpha_beta` is called with switched player perspective and decreased depth. So the depth corresponds to the number times the player switch turns and not to the moves that are made. We also implemented a mechanism to detect whether a branch led to the win of any player. In this case we stop the recursion so no time is wasted on this branch.

In the calculation of evaluation function uses two magic constants:

First: If one player won, we return a large constant of 1000000 if we won or -1000000 if we lost. This constants should be large enough to exceed every evaluation value generated in a non-final case.

Second: The seeds on the enemy side are counted and multiplied by 0.1 and then subtracted from the evaluation. This part of the evaluation should be seen as a tie breaker for two moves that otherwise would yield (nearly) identical differences in stored seeds. As a tie breaker it should not influence the evaluation function too strongly. The factor 0.1 was empirically proven to be useful.

7.8. yahiko (224480)

Idea

Simple minmax and complex evaluation function based on game stage. More importance to capture moves at later stages of game. Game stage decided based on number of seeds in house.

Minmax algorithm with complex evaluation function. The agent was written in Java.

7.9. ai-1 (224535)

Idea

Minimax Search algorithm with Alpha-Beta Pruning

Implementation

1. Run Minimax Alpha-Beta Pruning with increasing depth (2, 3, 5, 8, ...)
2. The moves are in descending order with respect to the evaluation score
3. Evaluation score is a linear combination of:
   - Store Advantage
   - Stone Differences
   - Capture Potential and Loss Potential
   - Extra bonus if agent can move again

7.10. ku81tuli;uj59ames;ez03odak (224684)

Idea

While having a minimax search with alpha-beta pruning and iterative deepening search, the main effort has been put into implementing a suitable evaluation function, like suggested in the paper. Simple move ordering techniques like searching for bonus moves first have also been tried out, but the algorithm has not shown significant improvements compared to this basic version. Therefore, we decided to submit the original version without move ordering.

Implementation

The agent has been implemented in python using the provided `kgp.py` file for the connection.

7.11. TheBestAgent (224715)

Idea

The agent uses the minmax algorithm with a custom evaluation function.

The evaluation function calculates the utility based on the number of seeds on each side of the board. It adds a quarter of the total seeds on the `SOUTH` side and the number of seeds in the `SOUTH` store to the utility. It then subtracts a quarter of the total seeds on the`NORTH` side and the number of seeds in the `NORTH` store from the utility.

The use of 0.25 as a multiplier makes sure that the AI values having seeds in its own side of the board, but not as much as having seeds in its store. This is is a ‘‘long term‘‘ protection to prevent having too few seeds on the maximizing side and loosing due to no seeds left on the side.

Implementation

The agent was written in Python.

7.12. Agent of Chaos (224732)

Idea

We used the minmax algorithm with alpha-beta pruning and move ordering.

Implementation

The agent was written in Java.

7.13. Stickfish (224734)

Idea

Alpha Beta Pruning with iterative deepening

Move ordering

- Extra moves from right to left
- Steals from left to right, but only if there are no bonus moves
- Other moves from right to left

Apart from that we change the moveorder depending on the results of our searches by moving the best move to the front

Evaluation

- Simulate all possible extra moves from right to left (go down only one branch of extra moves)
- Check the steal granting the most points. We subtract -0.75 to the side currently moving because we already simulated so many moves for it if the sum of the seeds not yet in a store are smaller than 3 * `board_size` we give a bonus to the side with more seeds on their half (because they keep them if the game ends) Do not ask how we came up with the values for these things (hint: it was not a genetic algorithm)
- Simulating the moves makes the code of the evaluation function very difficult to read and we can not say for sure if everything is correct f. e. if a move ends the game we will not notice it

Simulating the moves in the evaluation function is not what you usually do because the search simulates all moves already. But we only go down one branch of the tree and do it much faster because we only track our side.

Apart from that We save the `getMoves` results in a hashmap using the previous moves (saved in one long) as key. This allows us to put the `bestMoves` in the front for more pruning.

In addition we change the depth depending on how promising the move is by adding i/2 to the current depth for a move ordered at position i.

All in all we used the given `MinMaxAgent` as a template and were heavily inspired by the winner of the last year. (It says ‘‘don’t plagiarize‘‘ in the template but the template given is just how minmax works)

Implementation

Java very similar to the given minmax template. We implemented our own game logic because we saw the winners of last year did. It is slightly faster than before but we are not 100% sure if everything works correctly. We also store moves as bytes to save space in the hashmap.

7.14. na05kymu;ce84pegu (224867)

Idea

An agent that implements a MinMax algorithm with Alpha-Beta pruning. We do move ordering with a shallow lookahead of 1 to help the pruning process. As evaluation function/heuristic we used the store difference between us and the opponent. We also check if at any point we have more than half the points and stop the search returning the move. Also if the opponent has more than half the points we do the same thing.

Implementation

Agent is written in python using the template provided by `https://www-cip.cs.fau.de/˜oc45ujef/ai/kalah/python.tar.gz`. It has kgp.py module implemented and ready for use to communicate with the server. Also agent.py which holds the search algorithm as well as the evaluation function. The agent is implemented as a generator that yields different search results with different depths.

7.15. jo43beqy (224931)

Idea

I tried to make a move tree which always expands (thinking on opponent time) and updates when the oponent makes a move to make irrelevant brachen inacitve. Which Node is expanded depends on prio which is combination of eval score and depth The idea was to make eval a dot product of the state with weights and use deep learning techniques to get the best values for that.

However i could not get that all to work in time so I actually did a basic minmax.

Implementation

Python similar to the provided template.

7.16. EdizDerBreite (224952)

Idea

Minmax with alpha-beta-pruning.

Implementation

The agent was written in `C++`, using the Python implemenation of the protocol and communicating over `stdin` / `stdout`.