Coursework 2: Adversarial Search on Nim

Alejandro Ayuso García (CID: 00940554)
Konstantinos Mitsides (CID: 01857560)

Department of Computing
MSc in Artificial Intelligence

November 2023

1 Implementation of adversarial search by Minimax on Nim

Adversarial search by Minimax has been implemented on Nim in main.py. To run the program, the user only needs to execute the file. A flag has been included as an argument in the play() method to allow the user to either play a computer (`automatic = False`) or let the computer play itself (`automatic = True`).

2 Implementation of Minimax with $\alpha - \beta$ pruning

The Minimax implementation also allows the user to run the game with or without $\alpha - \beta$ pruning through the use of a second flag as argument in the play() method. In order to run Nim with $\alpha - \beta$ pruning the user should set the flag `is_pruned = True` and to run it with the vanilla Minimax algorithm the user should set the flag `is_pruned = False`.

3 Impact of m, n and k parameters on Minimax

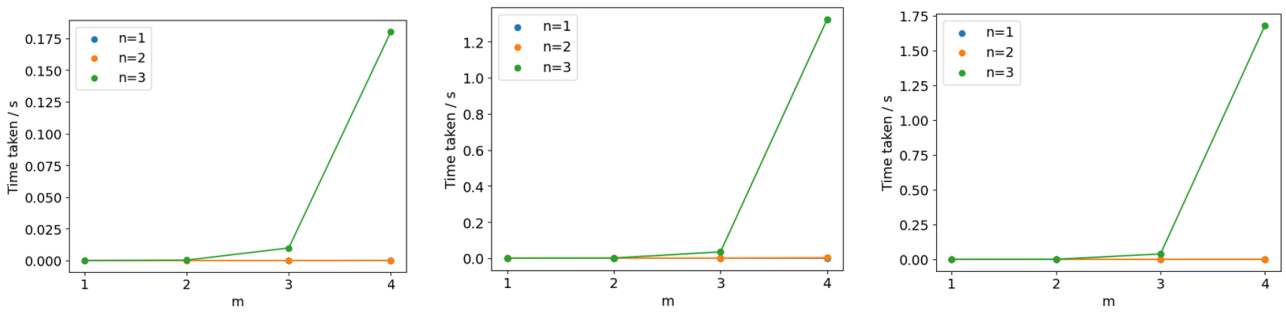
Minimax explores the game tree using a depth-first approach. As such, theory indicates that the time and space complexity of the algorithm are $\mathcal{O}(b^{d_{max}})$ and $\mathcal{O}(bd_{max})$, respectively, where b is the branching factor and d_{max} is the maximum depth of the game tree.

In Nim, players can only remove objects from a single heap in each turn, so we can describe the branching factor as: $b = kn$, where k is the maximum number of objects a player can remove from a single heap in a turn and n is the number of heaps. The maximum depth of the tree is described by $d_{max} = nm$, where m is the number of objects per heap, given that in the worst case scenario, players will remove one object from each heap until none remain. This means that the time and space complexity of a game of Nim are provided by the following equations: $\mathcal{O}((kn)^{nm})$ and $\mathcal{O}(kmn^2)$, respectively.

To explore the impact of parameters m, n and k on the time taken for action selection and the number of states explored, Nim was run multiple times with our implementation of the Minimax algorithm, varying the values of m, n and k between 1-4, 1-3 and 1-3, respectively. The exploration of wider value ranges was impractical due to the considerable computational cost associated and was therefore not pursued. However, the tested combinations, which are outlined in Tables 1 and 2, offer useful insights.

Time Taken / s				*m*			
				1	*2*	*3*	*4*
k	*1*	*n*	*1*	0.000000	0.000000	0.000000	0.000000
			2	0.000000	0.000000	0.000000	0.000099
			3	0.000000	0.000051	0.010001	0.167693
	2		*1*	0.000000	0.000000	0.000000	0.000000
			2	0.000000	0.000000	0.000108	0.001342
			3	0.000000	0.000564	0.035699	1.320705
	3		*1*	0.000000	0.000000	0.000000	0.000000
			2	0.000000	0.000000	0.000350	0.002282
			3	0.000000	0.001581	0.041551	1.737198

Table 1: Time taken for action selection by the implemented Minimax algorithm



(a) k = 1 (b) k = 2 (c) k = 3

Figure 1: Time taken by the Minimax algorithm implemented for action selection. Each point represents the mean of 10 runs. The line corresponding to n=1 (blue) is not visible as it is behind the line corresponding to n=2 (orange).

Figure 1 illustrates the impact of the three parameters on time taken for action selection. n has significantly larger impact on time taken than m, being consistent with the time complexity $\mathcal{O}((kn)^{nm})$. It is also observable in Table 1 that m has a greater impact than k.

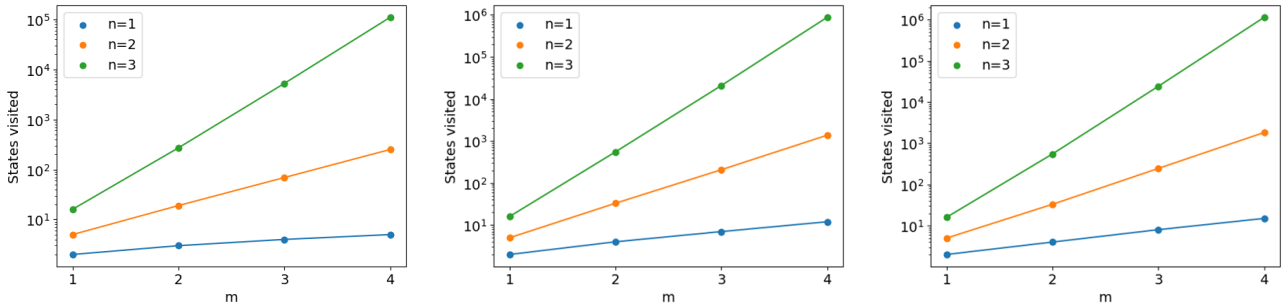
We can illustrate the differences in impact through one-step increases of each parameter from k=2, n=2, m=3:

- **With m**: the difference in time taken with respect to k=2, n=2, m=4 is (0.001342 - 0.000108 =) 0.001234, a 1140% increase
- **With n**: the difference in time taken with respect to k=2, n=3, m=3 is (0.035699 - 0.000108 =) 0.035591 (a 33,000% increase)
- **With k**: the difference in time taken with respect to k=3, n=2, m=3 is (0.000350 - 0.000108 =) 0.000242, a 224% increase

This is to be expected as n affects both the branching factor and the maximum depth of the game tree. m is observed to have a greater impact than k, indicating that maximum depth has greater importance than branching factor for time complexity. This behaviour matches the description of time complexity of depth first search $\mathcal{O}((kn)^{nm})$.

States visited				*m*			
				1	*2*	*3*	*4*
k	*1*	*n*	*1*	2	3	4	5
			2	5	19	69	251
			3	16	271	5248	110251
	2		*1*	2	4	7	12
			2	5	33	207	1369
			3	16	550	20761	871006
	3		*1*	2	4	8	15
			2	5	33	245	1827
			3	16	550	24136	1140793

Table 2: States visited for action selection by the implemented Minimax algorithm



(a) k = 1 (b) k = 2 (c) k = 3

Figure 2: States visited by the Minimax algorithm implemented for action selection. Each point represents the mean of 10 runs

Figure 2 illustrates the impact of m, n and k on the number of states visited by the algorithm during action selection on a logarithmic y scale. Again, we observe that n has a greater impact than m, and m has a greater impact than k on the number of visited states. Similarly to above, we can illustrate this by looking at the impact of one-step increases from k=2, n=2, m=3:

- **With m**: the difference in states visited with respect to k=2, n=2, m=4 is (1369 - 207 =) 1162, a 561% increase
- **With n**: the difference in states visited with respect to k=2, n=3, m=3 is (20761 - 207 =) 20554 (a 9,900% increase)
- **With k**: the difference in states visited with respect to k=3, n=2, m=3 is (245 - 207 =) 38, a 18% increase

n has significantly greater impact than either m or k, as would be expected given its power of 2 in $\mathcal{O}(kmn^2)$. k and m would be expected to have similar impacts since the former affects the branching factor and the latter affects the maximum depth of the game tree. However, since big O notation becomes a more accurate description of behaviour when parameters are sufficiently large, the larger impact of m might be explained by the low values of the parameters in this experiment. It must be noted however that k only has an impact on the number of states visited when m is greater than k, given that when the opposite applies the player cannot make use of the greater number of objects it can remove from a given heap since there are only m objects at most in it at any point in the game.

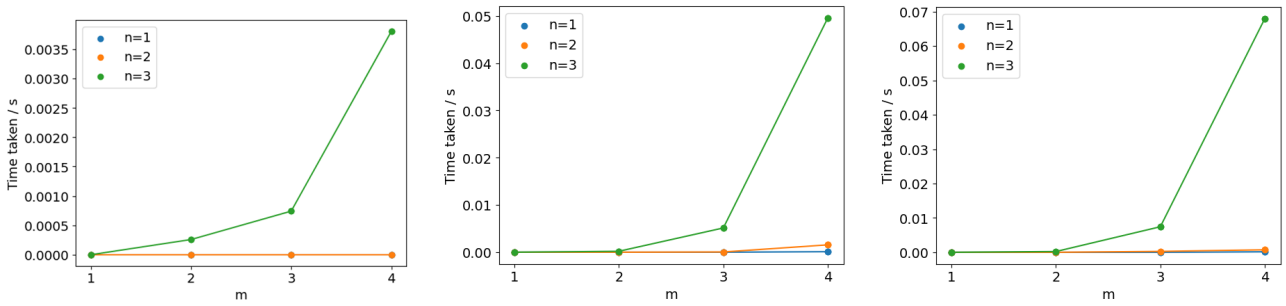
4 Impact of m, n and k parameters on Minimax with $\alpha - \beta$ pruning

$\alpha - \beta$ pruning permits ignoring nodes that are guaranteed to yield worse results than the current best. This addresses the significant computational cost associated with the vanilla Minimax algorithm without affecting whether the player wins or loses. $\alpha - \beta$ pruning effectively reduces the branching factor to $\sqrt{b}$ in the best case scenario - the extent of the reduction will depend on the order of the actions and the utilities these lead to -, rendering the time and space complexities of Minimax with $\alpha - \beta$ pruning $\mathcal{O}(b^{\frac{d_{max}}{2}})$ and $\mathcal{O}(\sqrt{b}d_{max})$, respectively. Based on the definitions of the branching factor and maximum depth above, these become: $O((kn)^{\frac{nm}{2}})$ and $O((k^{\frac{1}{2}}n^{\frac{3}{2}}m))$.

The same parameter value ranges were explored as in section 3 to facilitate the assessment of the impact of $\alpha - \beta$ pruning on performance.

Time Taken / s				*m*			
				1	*2*	*3*	*4*
k	*1*	*n*	*1*	0.000000	0.000000	0.000000	0.000000
			2	0.000000	0.000000	0.000000	0.000000
			3	0.000000	0.000259	0.000743	0.003806
	2		*1*	0.000000	0.000000	0.000000	0.000112
			2	0.000000	0.000000	0.000050	0.001554
			3	0.000000	0.000199	0.005131	0.049548
	3		*1*	0.000000	0.000000	0.000000	0.000151
			2	0.000000	0.000000	0.000277	0.000734
			3	0.000000	0.000233	0.007438	0.067938

Table 3: Time taken for action selection by the implemented Minimax with $\alpha - \beta$ pruning algorithm

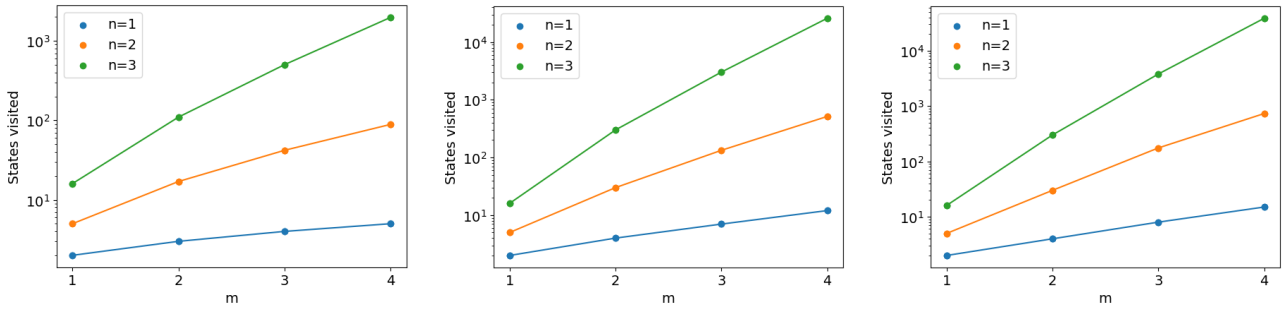


(a) k = 1 (b) k = 2 (c) k = 3

Figure 3: Time taken by the Minimax with $\alpha - \beta$ pruning algorithm implemented for action selection. Each point represents the mean of 10 runs

States visited				m			
				1	2	3	4
k	1	n	1	2	3	4	5
			2	5	17	42	89
			3	16	110	503	1981
	2		1	2	4	7	12
			2	5	30	133	514
			3	16	300	3017	25885
	3		1	2	4	8	15
			2	5	30	175	732
			3	16	300	3784	38418

Table 4: States visited for action selection by the implemented Minimax with $\alpha - \beta$ pruning algorithm



(a) k = 1 (b) k = 2 (c) k = 3

Figure 4: States visited by the Minimax with $\alpha - \beta$ pruning algorithm implemented for action selection. Each point represents the mean of 10 runs

Similar relative impacts of the parameters were observed on the time taken and states visited for action selection, with n remaining the most impactful of all three, and k having a greater impact than m.

We can illustrate the differences in impact on time taken through one-step increases of each parameter from k=2, n=2, m=3:

- **With m**: the difference in time taken with respect to k=2, n=2, m=4 is (0.001554 - 0.000050 =) 0.001504, a 3000% increase
- **With n**: the difference in time taken with respect to k=2, n=3, m=3 is (0.005131 - 0.000050 =) 0.005081 (a 10,200% increase)
- **With k**: the difference in time taken with respect to k=3, n=2, m=3 is (0.000277 - 0.000050 =) 0.000227, a 454% increase

Similarly, we can illustrate the impact of each parameter on the number of states visited by considering one-step increases from k=2, n=2, m=3:

- **With m**: the difference in states visited with respect to k=2, n=2, m=4 is (514 - 133 =) 381, a 286% increase
- **With n**: the difference in states visited with respect to k=2, n=3, m=3 is (3017 - 133 =) 2884 (a 2,170% increase)
- **With k**: the difference in states visited with respect to k=3, n=2, m=3 is (175 - 133 =) 42, a 32% increase

This behaviour is in line with the powers to which each parameter is raised in the equation of time complexity for $\alpha - \beta$ pruning: $O((k^{\frac{1}{2}} n^{\frac{3}{2}} m))$.

The improvement in performance with $\alpha - \beta$ pruning in terms of time and space complexity is significant, as seen in tables 1 and 2, and tables 3 and 4, respectively. If we consider the results for setting the parameters m=4, k=3 and n=3, the most complex combination considered, the difference is clear. Without $\alpha - \beta$ pruning the time taken for action selection was 1.74s, whereas with $\alpha - \beta$ pruning, this dropped to 0.0679s, a reduction in time of 96%. Similarly for the number of states visited, without $\alpha - \beta$ pruning, 1,140,793 states were visited, whereas with it only 38,418, a reduction of 97%.