COM2108 Functional Programming Assignment

Results

Experimental Weight Training

For *Tactic* **playOptimal** used by **optimalPlayer** each move is given a score, based off the points given from playing the domino and the average score the opponent can get based off the remaining dominoes in play and the new board created. This ratio can be given a weight association to value the points earned and potential points given to the opponent. To discover the best weight for said ratio, I decided to experiment by changing this weight and seeing what performed best.

I decided to test the player against the two opponents on a variation of five different seeds. These seeds were kept constant across both opponents in order to keep the test fair. The players I choice to test against were **scorePlayer** and **defensivePlayer**, as these have two varying tactics. I then tabulated the number of wins **optimalPlayer** obtained out of the 100 and calculated an average.

Player	Seed	Weight (w)				
		1	2	3	4	5
	17273	52	56	63	55	58
	32134	53	61	56	57	55
	43932	59	55	55	63	62
	83983	63	64	65	68	63
	90028	51	51	54	63	64
scorePlayer	**Average:**	**55.6**	**57.4**	**58.6**	**61.2**	**60.4**
	17273	89	93	92	92	91
	32134	86	89	93	88	88
	43932	88	92	93	94	96
	83983	91	90	94	93	88
	90028	91	96	94	91	92
defensivePlayer	**Average:**	**89**	**92**	**93.2**	**91.6**	**91**

We can infer from this that the **optimalPlayer** is much stronger against the **defensivePlayer**, this may be since the **playDefensive** tactic is less effective (this will be analysed in final testing). Evaluating the table, we can also see that when the weight is increased, the **optimalPlayer** performance will increase against **scorePlayer**, but decrease against **defensivePlayer.** This must be due to how the weight affect the output, increasing the weight means defensive plays are given more priority, helping defend against the **scorePlayer**. During final testing **optimalPlayer** will be tested against all other players, this means I should pick the best overall performing weight. Here I decided to test weights falling between 3 and 4 as these came out the strongest.

Player	Seed	Weight (w)				
		3	3.25	3.5	3.75	4
scorePlayer	17273	63	59	57	57	55
	32134	56	54	52	55	57
	43932	55	59	60	58	63
	83983	65	65	66	66	68
	90028	54	54	54	59	63
	Average won:	58.6	58.2	57.8	59	61.2
defensivePlayer	17273	92	92	92	94	92
	32134	93	90	92	90	88
	43932	93	97	94	94	94
	83983	94	93	93	96	93
	90028	94	94	95	94	91
	Average won:	93.2	93.2	93.2	93.6	91.6

After additional testing, I felt it would be beneficial to use a weight of 4 as the drop off for going down to a weight of 3.75 against **scorePlayer** would not be worth the gain of effectiveness for when against **defensivePlayer.** With the **optimalPlayer** scoring much higher against the **defensivePlayer** I also decided it was more important to prioritise performance against the more challenge opponent **scorePlayer**. If was to try and further improve the performance, I would need to test these variations of weight values against a larger pool of opponent player types.

Final Results

To evaluate my player designs I will go through and test all players against each other. For this I will use a variation of seeds that will stay constant throughout testing, and compute 1,000 games to ensure the data is accurate. Each table following represents a player's results against all other players.

scorePlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	966	975	961	966	971	966	966	977	**968.5**
scorePlayer	-	-	-	-	-	-	-	-	**-**
defensivePlayer	873	871	876	879	890	873	863	862	**873.375**
tacticalPlayer	481	461	454	454	490	478	475	463	**469.5**
optimalPlayer	400	394	398	385	393	400	393	386	**393.625**
prolongPlayer	491	457	486	491	503	497	458	477	**482.5**
nearWinPlayer	414	414	411	412	424	408	385	401	**408.625**
blockPlayer	864	879	864	868	871	881	860	859	**868.25**
							Total average:		**637.7679**

defensivePlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	862	888	886	859	888	881	901	871	**879.5**
scorePlayer	127	129	124	121	110	127	137	138	**126.625**
defensivePlayer	-	-	-	-	-	-	-	-	**-**
tacticalPlayer	93	92	95	99	123	100	115	98	**101.875**
optimalPlayer	50	58	59	51	73	64	68	48	**58.875**
prolongPlayer	107	115	121	122	127	107	97	132	**116**
nearWinPlayer	77	95	83	93	110	98	90	111	**94.625**
blockPlayer	529	501	531	511	540	498	531	463	**513**
								Total average:	**270.0714**

tacticalPlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	984	980	972	966	972	977	979	978	**976**
scorePlayer	519	539	546	546	510	522	525	537	**530.5**
defensivePlayer	907	908	905	901	877	900	885	902	**898.125**
tacticalPlayer	-	-	-	-	-	-	-	-	**-**
optimalPlayer	418	431	403	430	423	444	425	426	**425**
prolongPlayer	522	513	516	507	517	527	503	520	**515.625**
nearWinPlayer	445	448	468	446	460	439	450	451	**450.875**
blockPlayer	897	902	905	907	896	891	886	914	**899.75**
							Total average:		**670.8393**

optimalPlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	992	995	990	991	991	993	995	995	992.75
scorePlayer	619	610	610	579	583	598	613	607	602.375
defensivePlayer	937	936	927	947	932	938	927	946	936.25
tacticalPlayer	575	559	574	569	576	568	576	569	570.75
optimalPlayer	-	-	-	-	-	-	-	-	-
prolongPlayer	598	570	615	588	576	584	604	567	587.75
nearWinPlayer	523	558	550	532	527	523	539	558	538.75
blockPlayer	994	938	931	945	913	946	929	936	941.5
							Total average:		738.5893

prolongPlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	961	979	960	971	965	977	969	967	**968.625**
scorePlayer	509	543	514	509	497	503	542	523	**517.5**
defensivePlayer	893	885	879	878	873	893	903	868	**884**
tacticalPlayer	478	487	484	493	483	473	497	480	**484.375**
optimalPlayer	403	411	401	393	423	404	390	392	**402.125**
prolongPlayer	-	-	-	-	-	-	-	-	**-**
nearWinPlayer	428	442	445	434	441	432	413	424	**432.375**
blockPlayer	871	880	880	888	885	886	869	884	**880.375**
								Total average:	**652.7679**

nearWinPlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	996	993	987	994	992	990	992	992	**992**
scorePlayer	586	586	589	588	576	592	615	599	**591.375**
defensivePlayer	923	905	917	907	890	902	910	889	**905.375**
tacticalPlayer	555	552	532	554	540	561	550	549	**549.125**
optimalPlayer	433	466	483	469	460	459	472	445	**460.875**
prolongPlayer	572	558	555	566	559	568	587	576	**567.625**
nearWinPlayer	-	-	-	-	-	-	-	-	**-**
blockPlayer	906	906	896	908	921	923	890	902	**906.5**
								Total average:	**710.4107**

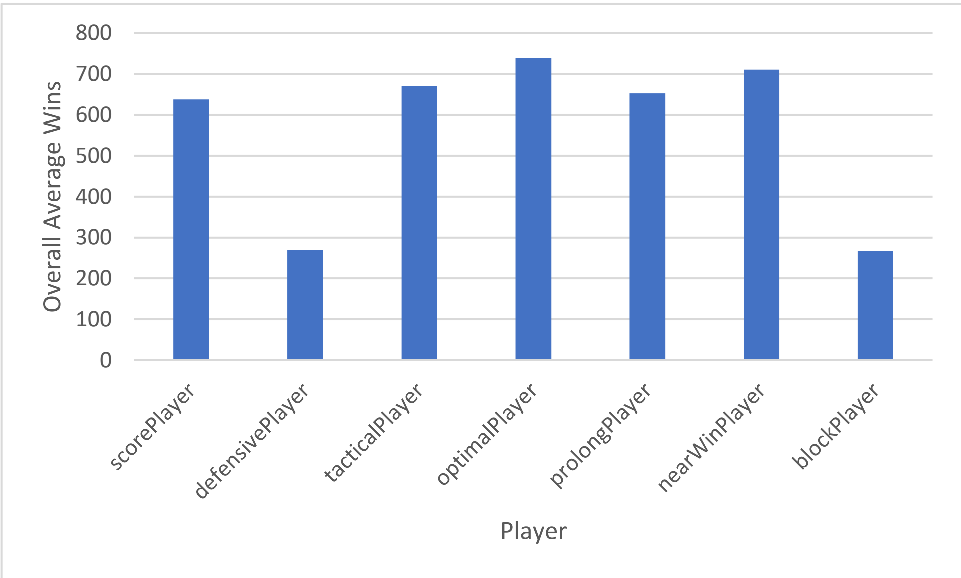
blockPlayer results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	851	872	883	849	886	872	879	886	**872.25**
scorePlayer	136	121	136	132	129	119	140	141	**131.75**
defensivePlayer	471	499	469	489	460	502	469	537	**487**
tacticalPlayer	103	98	95	93	104	109	114	86	**100.25**
optimalPlayer	64	61	55	55	57	71	70	60	**61.625**
prolongPlayer	129	120	120	112	115	114	131	116	**119.625**
nearWinPlayer	94	94	104	92	79	77	110	98	**93.5**
blockPlayer	-	-	-	-	-	-	-	-	**-**
								Total average:	**266.5714**

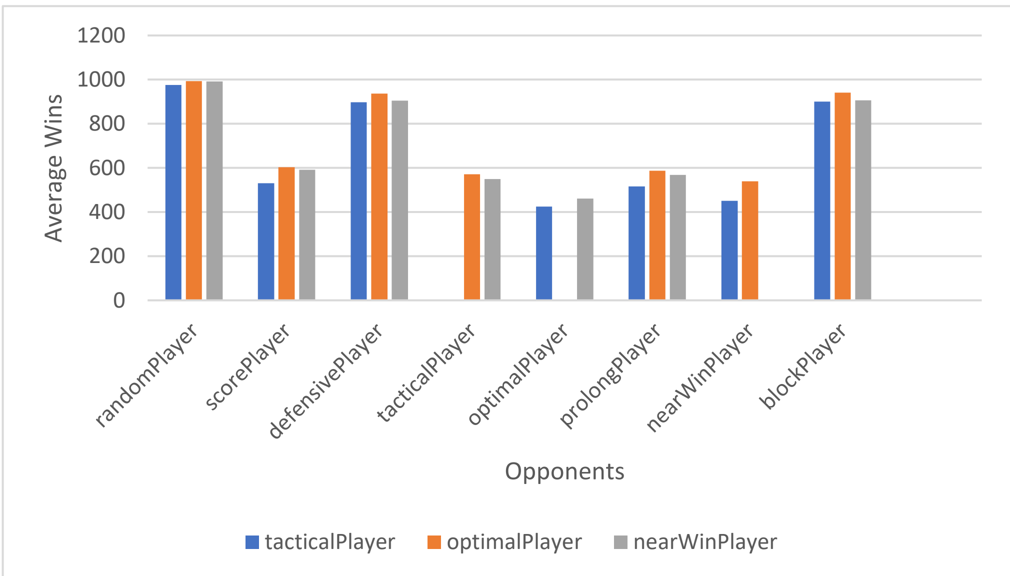
To further help visualise the result data, I compiled the average scores into one single matrix table. This makes it easier to compare two players, as well as see which player is strongest. Averages have been reduced to two decimal places where required.

	scorePlayer	defensivePlayer	tacticalPlayer	optimalPlayer	prolongPlayer	nearWinPlayer	blockPlayer
randomPlayer	968.5	879.5	976	989.38	968.63	992	872.25
scorePlayer		126.63	530.5	606.38	517.5	591.38	131.75
defensivePlayer	873.37		898.13	941.13	884	905.38	487
tacticalPlayer	469.5	101.88		575	484.38	549.13	100.25
optimalPlayer	393.63	58.88	425		402.13	460.88	61.53
prolongPlayer	482.5	116	515.63	597.88		567.63	119.63
nearWinPlayer	408.63	94.63	450.88	539.13	432.38		93.5
blockPlayer	868.25	513	899.75	938.38	880.38	906.5	
Total	**637.77**	**270.07**	**670.84**	**741.04**	**652.77**	**710.41**	**266.57**

A graph representing total average wins for each player



A graph representing the average win breakdown for top three performing players



Final Analysis

After completing the final testing, it became apparent that the top three performing players were **optimalPlayer, nearWinPlayer,** and **tacticalPlayer** (in order of efficiency). I then displayed the average score breakdown of these three players on a separate graph to further display how they performed. When against the **randomPlayer**, all of the domino players in my implementation greatly outperformed it. This is due to **randomPlayer** using the first playable domino in its hand, putting it at a great disadvantage.

When looking at the overall performance of my players, I can see that **defensivePlayer** and **blockPlayer** performed a lot worse than the others. This must be due to the fact they both primarily use the **playDefensive** tactic, which doesn't seem to be very efficient. I believe the main problem with this tactic is the fact it tries to capture what dominoes the opponent may have in their hand by using the dominoes which have been played and ones remaining in the players hand. This strategy is flawed as there are 28 dominoes in the set, but only a total of 18 in game, meaning the remaining dominoes technique is not very effective at capturing what dominoes the opponent may have. A way to improve this tactic would be to use the knowledge of what the opponent knocks on to work out which pips the opponent does not have in their hand and this may greatly reduce the uncertainty of what the opponent can play.

From analysing the second graph, I can infer that **optimalPlayer** outperformed the other two top players while against every other player, expect **randomPlayer** where **nearWinPlayer** outperforms it. I suspect this has occurred as the play **optimalPlayer** uses is affected by how much the opponent can potentially score, whereas **nearWinPlayer** will just go for points and winning plays. Trying to play more defensive against **randomPlayer** is a less efficient strategy as their moves are random so even if they can score high, it is less likely they will due to their play being the first playable domino in their hand.

Overall, I feel I have managed to create an array of smart domino players, with my top three scoring performing well against an array of opponents. My top performing player, **optimalPlayer**¸ managed to score well across the board so I believe it is a strong implementation of a smart domino player. However, I feel due to the fact optimal player only implements one tactic, it could be further adapted to improve its capabilities.

Building A Stronger Player

To improve on my current top scoring player, **optimalPlayer**, I will be creating a new player that uses the same **playOptimal** tactic, as well as using the **nearWin** tactic that assisted the **nearWinPlayer** in scoring so highly. I have also elected to create a additional improved player that will also implement the **blockWin** tactic, as although the **blockPlayer** did not perform well on the final result it because apparent that the **playDefensive** tactic was not strong so this may have been the cause. Following are the results of my two new improved players:

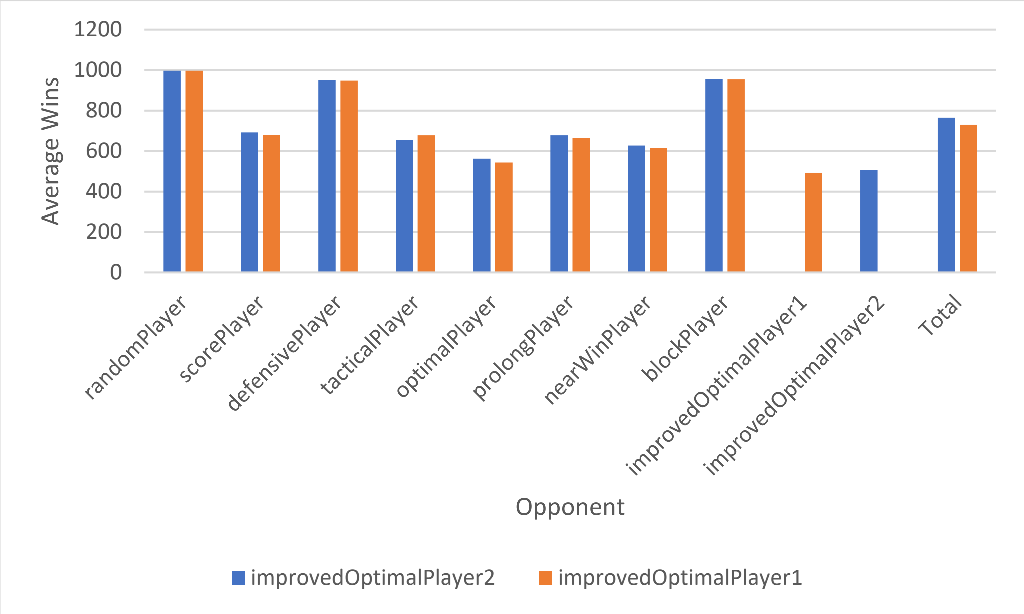
improvedOptimalPlayer1 results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	998	997	999	997	995	998	998	999	997.625
scorePlayer	687	708	695	684	681	697	692	692	692
defensivePlayer	954	951	942	963	944	953	942	959	951
tacticalPlayer	634	660	659	640	674	663	670	641	655.125
optimalPlayer	555	601	564	553	544	573	565	539	561.75
prolongPlayer	684	679	692	679	662	671	683	671	677.625
nearWinPlayer	609	661	618	632	618	614	629	635	627
blockPlayer	960	958	949	964	947	957	949	962	955.75
improvedOptimalPlayer1	-	-	-	-	-	-	-	-	-
improvedOptimalPlayer2	517	458	519	529	524	486	506	514	506.625
							Total average:		764.7344

improvedOptimalPlayer2 results:

Player	Seed								Average
	11232	32453	44322	49873	65421	76654	79987	91173	
randomPlayer	998	997	998	997	997	999	998	999	997.875
scorePlayer	679	700	692	664	664	671	687	680	679.625
defensivePlayer	954	946	940	961	936	952	937	955	947.625
tacticalPlayer	610	642	645	633	954	639	654	642	677.375
optimalPlayer	523	580	542	533	523	562	538	541	542.75
prolongPlayer	667	665	666	668	660	666	669	655	664.5
nearWinPlayer	598	642	614	616	607	609	621	627	616.75
blockPlayer	962	955	951	964	943	953	943	962	954.125
improvedOptimalPlayer1	483	542	481	471	476	514	494	486	493.375
improvedOptimalPlayer2	-	-	-	-	-	-	-	-	-
							Total average:		730.4444

Graph showing breakdown of average wins on improved players



From my additional testing I can see the newly created players are a definite improvement, giving the new highest averages amongst all pre-existing players. They both almost scored a near perfect 1000 wins against the **randomPlayer,** an impressive result. When comparing the two, **improvedOptimalPlayer1** came out on top when against the majority of oppositions, clearly demonstrating that it is the smartest player I have managed to create. However, **improvedOptimalPlayer2** has also shown to be another smart player, scoring just under what the former managed to achieve. Showing that both players simulate a good understanding on how to successfully win dominoes '5s and 3s'. The only difference between the players is that the latter makes use of the **blockWin** tactic, however this seems to put it at a disadvantage. I theorise this may be occurred from the evaluation of remaining dominoes not accurately representing what dominoes the opponent has in their hands (as before mention, this could potentially be improved by using knocking information to the player advantage).

To conclude, after designing and implementing an array of domino players, I managed to evaluate the successful player strategies and tactics in order to improve upon my current strongest player. This led me to complete additional testing which proved my newly created players had a greater understanding of the dominoes game '5s and 3s'. I have also discovered within the scope of my assignment that using an optimal scoring method to assess both the highest scoring and best defensive plays as well as using a near win tactic to secure 61 or 59 points remains the strong approach for a computer player in the game of '5s and 3s'.