**Section One – Considering Critical Cases**

**Introduction**

**Considering A Win**

For my base player, I begun by taking the logic of the HSD Player and creating a tactic based around this strategy. I then aimed to capture some of the endgame logic, by specifically seeking out PossMoves from the hand which would result in a win (whereas the HSD Player just plays random dominoes until it secures that win or knocks).

|  |  |
| --- | --- |
| **Opponent** | hsdPlayer |
| **Win : Loss** | 545 : 455 |
| **Percentage Win** | 54.5% |

From the evaluation above, should we assume the probability of a HSD Player winning against another HSD Player be 50%, that the new logic which seeks a win condition improves performance by 4.5% over the base evaluation HSD Player.

**Considering Opponent’s Win**

Next, I decided to consider the conditions in which the opponent can win the game, as it is perhaps possible to block the opponent from winning towards the end of the game, based on their hand and stitching tactics. As a result, I implemented the stopOpponentWin tactic. The results of implementing this are shown below.

|  |  |  |
| --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer |
| **Win : Loss** | 545 : 455 | 469 : 531 |
| **Percentage Win** | 54.5% | 46.9% |
| **Percentage Improvement Over winPlayer** | 0.0% |  |

From the test above the first iteration of the blocking logic shows no increase in performance over the previous iteration of the player. Following further examination, I decided this could be due to the face we do not know exactly what is in the opponent’s hand, only what isn’t already on the board or in the player’s hand. Therefore, I decided to only protect against likely wins, rather than against everything, because at this point in the game, we also want to maximise our own player’s score.

|  |  |  |
| --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer |
| **Win : Loss** | 582 : 418 | 505 : 495 |
| **Percentage Win** | 58.2% | 50.5% |
| **Percentage Improvement Over winPlayer** | 3.7% |  |

By playing blocks against wins which are more aggressive, performance has significantly improved over the previous iteration. There is now a significant improvement in the performance of the player, and it consistently wins in most games against both opponents.

**Playing (5,4) On Initial Board**

(5,4) is the best domino which can be played on the initial board, as it scores three but the opponent can only score two on the following turn. This is a simple tactic with little logic and the improvement in performance gained from this is expected to be marginal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer |
| **Win : Loss** | 565 : 435 | 508 : 492 | 479 : 521 |
| **Percentage Win** | 56.5% | 50.8% | 47.9% |
| **Percentage Improvement Over blockPlayer** | -1.7% | 0.3% |  |

Surprisingly, playing (5,4) on the initial board isn’t a good strategy and unexpectedly results in poorer performance than the Blocking Player. Therefore, it will not be included in the strategy for subsequent players.

**The FiftyNine Tactic**

The FiftyNine tactic is designed to make the player’s score up to fifty-nine as there are more ways to score 2 from 59 to reach the score limit than there are ways to score 1 from 60, 3 from 58. Naturally, as this sets up a win it should be evaluated high up in the list of critical tactics, but should be considered less important than blocking against an opponent win.

The tactic should also account for a couple of key criteria:

* If the player’s score is already 59, the tactic shouldn’t return anything, as otherwise the tactic will return lots of values which keep the player’s score at 59.
* The player does not have a domino which scores two in their hand to follow up the 59 making play. They tactic should deal with this case, only returning the 59 scoring domino if a following domino delivers a win.

Following the addition of this tactic, the player performs as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer |
| **Win : Loss** | 582 : 418 | 506 : 494 | 483 : 517 |
| **Percentage Win** | 58.2% | 50.6% | 47.9% |
| **Percentage Improvement Over blockPlayer** | 0.0% | 0.1% |  |

Trying to make the player’s score up to fifty-nine seems to have resulted in the player winning a single extra game against the win player. However, this player performs worst when pitted against the blocking player. This tactic isn’t adding meaningfully to the performance of the program and once again, the blockPlayer performs better.

**A Better High Scoring Tactic**

The current high scoring tactic has no look-ahead concept. It is pointless playing the highest scoring domino if the opponent can score higher on their next turn. As a result, the player should try and play a safe high scoring domino before falling back to more reckless dominoes. There are two implementations of this to consider:

* The player only plays a HSD if the opponent does not score higher on their next turn because of the play, at the new end.
* The player only plays the HSD if their total score remains higher on the next play, that the total score of the opponent after their next turn.

In the case of the first implementation, the program scores as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer |
| **Win : Loss** | 632 : 368 | 557 : 443 | 537 : 463 |
| **Percentage Win** | 63.2% | 55.7% | 53.7% |
| **Percentage Improvement Over blockPlayer** | 5.0% | 5.2% |  |

By enhancing the HSD function in this way, the first implementation of the better HSD tactic scores around 5% better against both the original HSD player, and the winPlayer. It also beats the blockPlayer in most games, making it the new best version of the player.

The second implementation does not work as well, even when logic is included to fall back to implementation one if the opponent starts ahead. This is because it is a greedy strategy and has less consideration of the rest of the game, sprinting ahead only to stall as the hand becomes more limited.

**Section 2 – Advisory Tactics**

In addition to the critical tactics, there are some advisory tactics which attempt to return good dominoes to play in the case that the critical tactics need not be used. For the initial implementation, the system will use a polling system to consider which domino is best to play. Dominos suggested by the tactics are counted and the most frequently suggested domino is played.

Trying to order advisory tactics logically may not be optimal, as the natural order isn’t explicit, as is the case with the critical tactics. For this implementation, the recklessHSD will still be the fallback tactic in the case all others fail.

**Initial Implementation**

The following implementation has implemented the playMajority, opponentStitch, knockOff and reckless HSD tactics in its advisory function. It takes the most popular domino amongst the tactics and plays it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer | safeHSDPlayer |
| **Win : Loss** | 639 : 361 | 564 : 436 | 548 : 452 | 530 : 470 |
| **Percentage Win** | 63.9% | 56.4% | 54.8% | 53.0% |
| **Percentage Improvement Over safeHSDPlayer** | 0.7% | 0.7% | 1.1% |  |

The polling system for advisory tactics seems to work well, increasing performance when compared to all prior iterations of the player. The player now wins in around 64% of games against a simple HSD Player.

**Experimenting with Alternative Selection Methods**

The polled player can potentially be improved via alternative selection methods, as right now the polled tactics do not care whether the dominoes they return advance the player’s score towards a win. We could implement this knowledge in the following ways:

* Play Aggressively – Return the highest scoring domino from the pack of suggested dominoes, considering the strategies and how they score primitively.
* Weighted Evaluation – Weight the merit of a domino on a function of (Times Suggested \* Domino Score)

Playing Aggressively:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer | safeHSDPlayer | polledPlayer |
| **Win:Loss** | 595:405 | 526:474 | 506:494 | 485:515 | 469:531 |
| **Percentage Win** | 59.5% | 52.6% | 50.6% | 48.5% | 46.9% |
| **Percentage Improvement Over polledPlayer** | -4.4% | -3.8% | -4.2% | -4.5% |  |

The vanilla polling strategy is more effective. This strategy is too greedy and playing in this way gives worse performance than polling. This strategy meant the aggressive player lost against some of the prior players, including the safeHSDPlayer.

Playing Weighted:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer | safeHSDPlayer | polledPlayer |
| **Win:Loss** | 657:343 | 577:423 | 547:453 | 529:471 | 526:474 |
| **Percentage Win** | 65.7% | 57.7% | 54.7% | 52.9% | 52.6% |
| **Percentage Improvement Over polledPlayer** | 1.8% | 1.3% | -0.1% | -0.1% |  |

These results present an issue because against the polled player, and the two most simplistic players, this evaluation strategy performs better. However, in the case of winPlayer and blockPlayer, this evaluation strategy won one less game, than the polledPlayer. I will continue to use the weighted strategy for evaluation, as when pitted against the polledPlayer (the most intelligent player explored so far), it has superior performance.

**Improving the Weighted Player**

The code used for the weighted player could be easily adapted to further tweak the weighting function such as, to also consider the function which returned a given domino in the weighting of a suggestion’s importance. For this implementation, I decided to stick with evaluating each advisory tactic with equal merit as informal tests showed that the performance gained from this would be neglegible.

**Section 3 – Tactic/Strategy Improvement**

This section will focus on steps which can be taken to improve the existing strategies and tactics which have been outlined above.

**Going Bust**

In some games, the player cannot win with an exact 61, so should instead strive to score a 60, instead of just playing the highest scoring domino repeatedly. This is part of the advisory tactics as by this point in the flow of execution for a player, we have established that the opponent cannot win the game on their current turn so the current player should instead try to maximise their own score, without going bust.

To implement this, I added extra conditions to the HSD Tactics to ensure the player doesn’t go bust, and the weightedAdvice function.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Opponent** | hsdPlayer | winPlayer | blockPlayer | safeHSDPlayer | polledPlayer | weightedPlayer |
| **Win:Loss** | 663:337 | 593:407 | 570:430 | 546:454 | 537:463 | 530:470 |
| **Percentage Win** | 66.3% | 59.3% | 57.0% | 54.6% | 53.7% | 53.0% |
| **Percentage Improvement Over Old weightedPlayer** | 0.6% | 1.6% | 2.3% | 1.7% | 0.5% |  |

Increasing the player’s awareness of going bust was a clear success. The player won (presumably) the same nine games it previously lost against all the four most primitive players. Performance of the overall player now sits at a healthy 66.6%.

**Section 4 – Conclusion**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opponent** | hsdPlayer | randomPlayer | winPlayer |
| **Win:Loss** | 663:337 | 985:15 | 593:407 |
| **Percentage Win** | 66.3% | 98.5% | 59.3% |

In conclusion, the player’s performance against the hsdPlayer has increased by 12.1% over the iterative development of the Player, when compared with the winPlayer.

Additionally, when compared to the performance of a random player against the hsdPlayer, percentage of wins in the final player is 62.3% higher than that of a random player, so overall, the dominoes player works reasonably well.

**Section 5 – Testing of Final Players**