An Analysis of Derived Belief Strategy's Performance in the 2005 Iterated Prisoner's Dilemma Competition

Tsz-Chiu Au
Department of Computer Science
University of Maryland, College Park, MD 20742 USA.
chiu@cs.umd.edu

Technical Report CS-TR-4756 / UMIACS-TR-2005-59

February 19, 2006

Abstract

We analyzed the performance of the Derived Belief Strategy (DBS) in the 2005 Iterated Prisoner's Dilemma Competition [1]. Our technique is to remove the scores of the agents submitted by the participants one by one in the computation of the average scores.

1 The Results of our Programs in the Competition

The 2005 Prisoner's Dilemma Competition consists of four competitions. The first competition, namely the Competition 1, is a re-run of the original experiment of Robert Axelrod that was held twenty years ago; the other competitions are similar to the Competition 1 except that they are run under different conditions and rules. Our strategies aimed for the second competition, the Competition 2, which is the same as the Competition 1 but has a noise level of 0.1. In the following, we shall only focus on the results of our programs in the Competition 2.

The Competition 2 consists of five runs. Each run is a round-robin tournament in which each program plays with every program once, including itself. There are a total of 165 programs in the Competition 2. Among them 8 programs are default programs provided by the organizer of the competition, while the remaining 157 programs are programs submitted by 36 different participants. Even though it is allowed to submit up to 20 programs, 29 participants submitted only one program. Nonetheless, four participants submitted 20 programs, one of them is the author of this report.

The default programs are ALLC (always cooperates), ALLD (always defects), GRIM (cooperates until the first defection of the other player, and thereafter it always defects), NEG (cooperate (or defect) if the other player defects (or cooperates) in the previous iteration), RAND (defects or cooperates with the $1/2$ probability), STFT (suspicious TFT, which is like TFT except it defects in the first iteration) TFT, and TFTT (Tit-for-two-Tats, which is like TFT except it defects only

after it receives two consecutive defections). They are well-known in the literatures of the iterated Prisoner's Dilemma.

We submitted twenty programs in the competition. They are the implementations of the three different strategies proposed by the author. This report presented the results of one of these strategies, the *Derivative Belief Strategy* (DBS). The other two are called Tat-for-Tat Improved (TFTI) and Learning of Opponent's Strategy with Forgiveness (LSF).

Like DBS, LSF is a strategy that learns the other player's strategy during the game. The difference between LSF and DBS is that LSF does not make use of the symbolic noise filtering but uses the discount frequency only to learn the other player's strategy. In addition, it incorporates a forgiveness strategy that decides when to cooperate when mutual defection occurs. There is only one program that makes use of LSF, and it is also called LSF. In the Competition 2, LSF is at around the 30'th places in three runs and around 70'th places in the other two runs. We believe the poor ranking of LSF is due to the deficiency of using discount frequency alone.

TFTI is a strategy based on a totally different philosophy than DBS's. It is not an opponent modeling strategy in the sense that it does not model the other player's behavior using a set of rules. Instead, it is a variant of TFT with a sophisticated forgiveness policy that aims at overcoming some of the shortcomings of TFT in noisy environments. As we shall see, the best implementations of TFTI is one of the top five programs in the Competition 2 (if we excludes the multiple copies of DBS). This paper focuses on DBS only, and we shall not discuss TFTI. We submitted ten programs of TFTI, each of them has a different set of parameters or different implementation. Among them TFTIm has the best performance.

Like TFTI, we have submitted nine programs of DBS, each of them has a different set of parameters or different implementation. The best DBS program is DBSz, which makes use of all features of DBS. Some other DBS's makes use ideas that have never discussed in this report. Nonetheless, it seems that these ideas do not greatly enhance their performance.

The other three participants who has submitted twenty programs are Wolfgang Kienreich, Jia-wei Li, and Perukrishnen Vytelingum. According to our analysis of the competition results, we recognize that they adopted a strategy called the *master-slave strategy*, that was first proposed by a team from the University of Southampton in England. A master-slave strategy is actually not a strategy for a single program but for a team of collaborating programs. One of the program in such a team is the *master*, and the remaining programs are *slaves*. The basic idea is that the master and slaves recognize each other through a series of moves at the beginning of a game through a predefined protocol. If a slave recognizes that the other player is the master, it will cooperate thereafter; conversely if the master recognizes that the other player is a slave, it continues to defect. Thus once the recognition is established, the master can earn as much points as possible in every following iteration at the expense of the slaves. Furthermore, if a slave find that the other player is not the master, it will try to minimize the score of the other player by defections.

We notice that these participants have selected one of the their 20 programs as a master, and the remaining are assigned to be slaves of their masters. Wolfgang Kienreich's master is CNGF (CosaNostra Godfather), and the slaves of CNGF are 19 copies of CNHM (CosaNostra Hitman). Jia-wei Li's master is IMM01 (Intelligent Machine Master 01), and the slaves of IMM01 are IMM01, IMM02, ..., IMM20, which are called Intelligent Machine Slave n for $n = 02, 03, \ldots 20$. Perukrishnen Vytelingum's master is BWIN (S2Agent1_ZEUS), and the slaves of BWIN are BLOS2, BLOS3, ..., BLOS20, whose names are the names of different Gods in anicent Greek.

By the time we wrote this report, we do not know what strategies other particpants, except

whose we mentioned above, used in their programs.

1.1 The Competition 2 and the Results

The Competition 2 consisted of five runs. Each run was a round-robin tournament in which each program plays with every program once, including itself. Each program will participate in 166 games in each run (the game that both players are the same program counted as two games). Each game consists 200 iterations. [1] The *point* a program gets in a game is the sum of all payoffs it earns in all 200 iterations. Hence, the maximum point one can get in a game is 1000, and the minium point is 0. The *total point* of an program in a run is the sum of the points of all games in all 166 games it participates in the run. The higher the total point of an program, the better the program plays. On the compeititon's website, there is a ranking for each of the five runs, each of them is ranked according to the total point of the programs in a run.

We define a *point profile* of a program in run r to be the set of points the program earns in run r. Suppose $\{P_1, P_2, \ldots, P_{165}\}$ be the set of all programs in the competition. Let $Point_r(i,j)$ and $Point'_r(i,j)$ are the points earned by P_i and P_j respectively in the game in which P_i is the Player 1 and P_j is the Player 2 in run r. The point profile of P_i in this run is $\{Point_r(i,j) : P_j\} \cup \{Point'_r(i,i)\}$. The total point of P_i in run r is the sum of all points in its point profile in run r. The *mean point* of P_i in a run r is equal to the average of points in its point profile. The rankings remain the same if we rank according to the mean points.

The *average points* of P_i against P_j is $A(i,j) = (\sum_{1 \le r \le 5} Point_r(i,j))/5$. Likewise, the average points of P_i against itself as the Player 2 is $A'(i,i) = (\sum_{1 \le r \le 5} Point'_r(i,i))/5$. Since each game is independent of each other no matter which run the game takes place, we deem that the average point is a good measure of the overall performance of P_i against P_j in the competition. The *average point profile* of P_i is $AvgProfile(i) = \{A(i,j) : P_j\} \cup \{A'(i,i)\}$. The *mean average point* is the average of all average points in $AvgProfile(i)$.

Table 1 shows the mean points in each runs of the five runs and their mean (i.e., the average points) of the top twenty-five programs when the programs are ranked by their average points. We submitted nine programs using DBS, and all of them were among the top twenty-five programs; they dominated top ten places in the list, except that it lost to BWIN and IMM01, the masters of two master-slave strategies. This phenomenon implies that the performance of DBS are insensitive to the parameters in the programs and the implementation details of an individual program. The same phenomenon happens to TFTI—nine out of ten programs using TFTI were ranked between the 11th place and the 25th place, and the last one was at the 29th place.

1.2 Mean Average Points versus the Number of Slaves Removed

We would like to study the effects of slaves to the result of the Competition 2. Our approach is to select a number of slaves and then exclude the points of the games that involve these slaves from the calculation of their average points, as if the slaves have never participated in the competition. As we gradually removed slaves from the calculations, we expect to see that mean average points of the other players increases, while the mean average points of the masters decreases. More precisely, suppose we want to remove a set of programs $S = \{S_1, S_2, \ldots, S_k\} \subseteq \{P_i : 1 \le i \le 165\}$, the

[1] An iteration is also called a period or a round in some literatures

Table 1: The mean points of the programs in each of the five runs in the Competition 2, and their mean average point. This table is ranked according to the mean points. Only top twenty-five programs are listed.

Rank	Programs	Participant	Mean Point					
			Run1	Run2	Run3	Run4	Run5	Average
1	BWIN	Perukrishnen Vytelingum	441.7	431.7	427.1	434.8	433.5	433.8
2	IMM01	Jia-wei Li	424.7	414.6	414.7	409.1	407.5	414.1
3	DBSz	Tsz-Chiu Au	411.7	405.0	406.5	407.7	409.2	408.0
4	DBSy	Tsz-Chiu Au	411.9	407.5	407.9	407.0	405.5	408.0
5	DBSpl	Tsz-Chiu Au	409.5	403.8	411.4	403.9	409.1	407.5
6	DBSx	Tsz-Chiu Au	401.9	410.5	407.7	408.4	404.4	406.6
7	DBSf	Tsz-Chiu Au	399.2	402.2	405.2	398.9	404.4	402.0
8	DBStft	Tsz-Chiu Au	398.4	394.3	402.1	406.7	407.3	401.8
9	DBSd	Tsz-Chiu Au	406.0	396.0	399.1	401.8	401.5	400.9
10	lowES-TFT_classic	Michael Filzmoser	391.6	395.8	405.9	393.2	399.4	397.2
11	TFTIm	Tsz-Chiu Au	399.0	398.8	395.0	396.7	395.3	397.0
12	Mod	Philip Hingston	394.8	394.2	407.8	394.1	393.7	396.9
13	TFTIz	Tsz-Chiu Au	397.7	396.1	390.7	392.1	400.6	395.5
14	TFTIc	Tsz-Chiu Au	400.1	401.0	389.5	388.9	389.2	393.7
15	DBSe	Tsz-Chiu Au	396.9	386.8	396.7	394.5	393.7	393.7
16	TTFT	Louis Clement	389.1	395.8	394.1	393.4	394.7	393.4
17	TFTIa	Tsz-Chiu Au	389.5	394.4	395.1	389.6	397.7	393.3
18	TFTIb	Tsz-Chiu Au	391.7	390.0	390.5	401.0	392.4	393.1
19	TFTIx	Tsz-Chiu Au	398.3	391.3	390.8	391.0	393.7	393.0
20	mediumES-TFT_classic	Michael Filzmoser	396.7	392.6	398.3	390.8	386.0	392.9
21	TFTIy	Tsz-Chiu Au	391.7	394.6	390.8	392.1	394.9	392.8
22	TFTId	Tsz-Chiu Au	395.6	393.1	388.8	385.7	391.3	390.9
23	TFTIe	Tsz-Chiu Au	396.7	391.1	385.2	388.2	393.5	390.9
24	DBSb	Tsz-Chiu Au	393.2	386.1	392.6	391.1	391.0	390.8
25	T4T	David Fogel	391.5	387.6	400.4	387.3	383.5	390.0

new mean average point of $P_i \notin S$ is

$$A(i|\neg S) = \frac{\sum_{1 \le r \le 5} \left\{ Point'_r(i,i) + \sum_{1 \le j \le 165 \text{ and } P_j \notin S} Point_r(i,j) \right\}}{5 \times (166 - k)}$$

Suppose we want to remove k BWIN's slaves, k IMM01's slaves, and k CNGF's slaves from the competition. Let S^{BLOS}, S^{IMS}, $S^{\text{CNHM}} \subseteq \{P_i : 1 \le i \le 165\}$ be the set of slaves of BWIN, IMM01, and CNGF, respectively. First, we randomly choose k slaves from each set of slaves and let them be $S^a_k \subseteq S^{\text{BLOS}}$, $S^b_k \subseteq S^{\text{IMS}}$, and $S^c_k \subseteq S^{\text{CHHM}}$, respectively. The new mean average point of P_i which is not a slave is $A(i|\neg(S^a_k \cup S^b_k \cup S^c_k))$.

This mean average point depends on the selection of the slaves to be removed. To minimize biases toward a particular set of slaves, we repeated the above procedure one hundred times with different selection of slaves, and obtained 100 mean averages points for each program for each value of k. Then we calculated the minimum, the maximum, and the average of these 100 mean average points using the data provided by the organizer of the competition, and plotted the mean averages against k as shown in Figure 1, for a number of top-twenty programs. We also show the minimum and the maximum mean average points of BWIN, IMM01, CNGF, and DBSz by using error bars.

Figure 1 shows that as the number of slaves decreases, the mean average points of all selected programs increases. The rate of increases of the non-master programs is much faster than the rate of increases of the masters. For instance, the mean average points of DBSz increases by 3.9 as k increases by 1, whereas the mean average points of BWIN increases only by 0.33. This implies that slaves does not cooperate with the non-master programs but some of the masters. The mean average points of a master increases because the slaves of different masters does not cooperate the master. Furthermore, the rate of increases are not constant; as the number of slaves decreases, the rate of increases of the mean average points increases. It is due to the fact that the pool of the programs is getting smaller as the slaves is removed.

We can see that the average of the mean average point of DBSz surpasses the IMM01's when $k \ge 3$ and the BWIN's when $k \ge 9$. Furthermore, the DBSz's mean average points becomes the highest when when $k \ge 9$. It is more certain when $k \ge 10$, as the minimum of the mean average point of DBSz is more than the maximum of the mean average point of IMM01 when $k \ge 10$.

1.3 Mean Average Points versus the Number of Our Programs Removed

In order to show that our programs do not act like the master-slave strategy, we conducted an analysis similar to the one in the previous section: we removed most of our programs one by one to see how it affects the mean average points of selected programs. The 18 programs to be removed are DBSa, DBSb, DBSd, DBSe, DBSf, DBSx, DBSy, DBSpl, DBStft, TFTIa, TFTIb, TFTIc, TFTId, TFTIe, TFTIf, TFTIx, TFTIy, TFTIz, LSF. These includes all programs we have submitted except DBSz and TFTIm.

We followed the same procedure in the previous section, and plotted the graph in Figure 2. As can be seen, the mean average points of most programs decrease as the number of our programs removed increases. This indicates that our programs have cooperated with them and increased their points on average. As opposed to the slaves in the master-slave strategies, our programs have positive influence to the programs submitted by other participants.

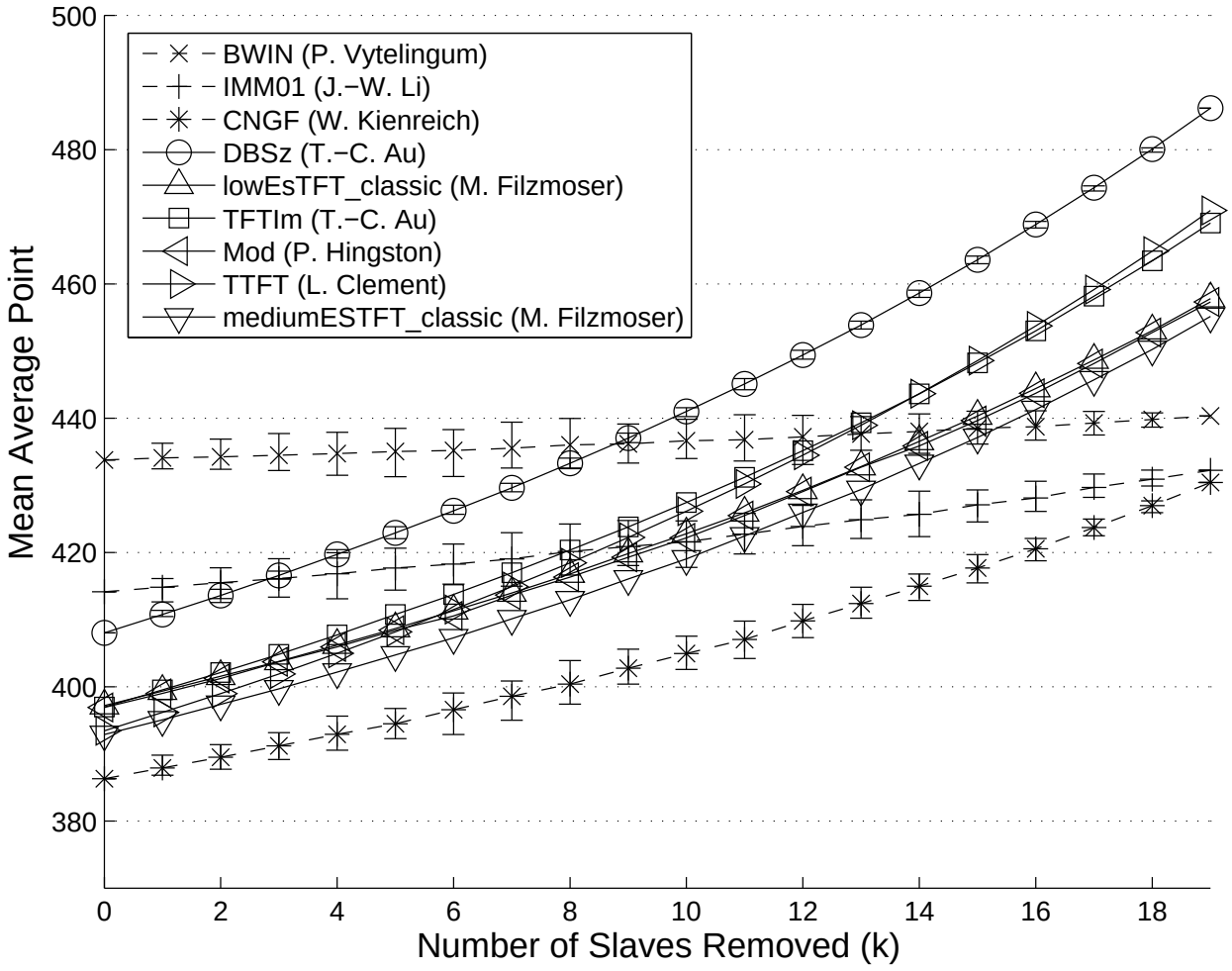

Figure 1: Average points of selected programs as the number of slaves decreases. We randomly selected k BWIN's slaves, k IMM01's slaves, and k CNGF's slaves, and removed them from the calculation of the mean average points, using the data provided by the organizer of the competition. We repeated the procedure 100 times and calculated the average of the mean average points against k, as shown in each data point in this graph. We also showed the maximum mean average points and the minimum mean average points of BWIN, IMM01, CNGF, and DBSz.

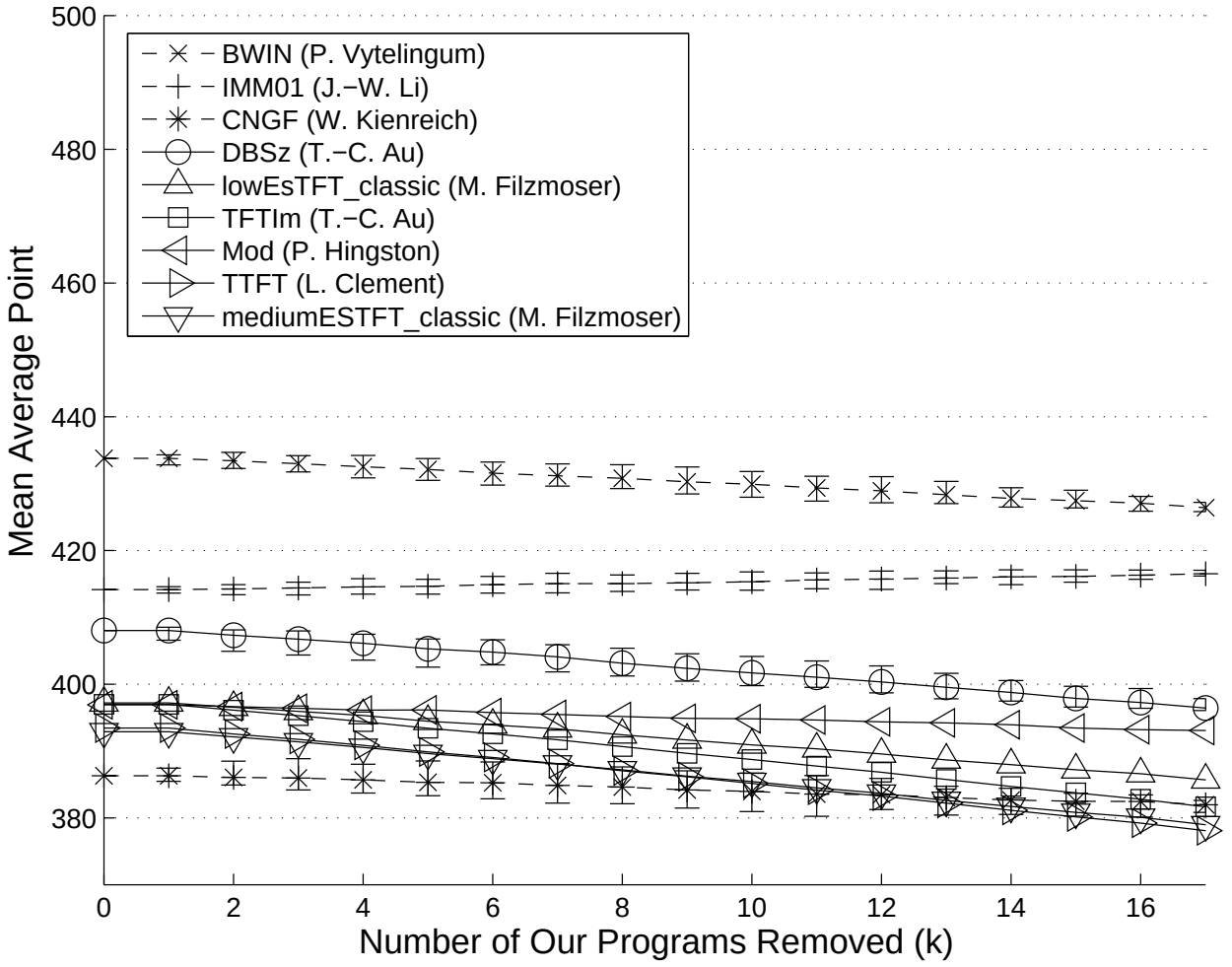

Figure 2: The mean average points of the selected programs as the number of our programs decreases. As in Figure 1, we randomly selected k programs from the set of our programs except DBSz and TFTIm, and removed them from the calculation of the mean average points. We can see that most of the selected programs have lower mean average points after the removal. It implies that the removed programs have cooperated with them and increased their points on average.

1.4 Distributions of Average Points

The mean average point of a program is a measure of the overall performance of the program in the competition. However, the mean average points give us little insight into why a program works better. Perhaps a better way to study a program's performance is to look into its average point profile to see the distribution of the average points.

We counted how many average points in $AvgPoint(i)$ that are fall into the interval $(5k - 2.5,\ 5k + 2.5]$ for each $0 \leq k \leq 200$, and then plot k against this number. This plot is called the *density plot* of the average points. Figure 3 and Figure 4 show the plots for selected programs (including some of the top twenty-five programs plus the masters and one of their slaves). Notice that we do not show the scale of the y-axis as we do not concern with the actual numbers in the intervals; all we care about is the *distribution* of average points.

In order to establish the link between programs P_j and the average points, we overlaid the density plots with *dot diagrams*, which indicates the average points of P_i against different groups of programs. We partitioned the set of all 165 programs into six groups: (1) the DBS programs, (2) the TFTI programs, (3) BWIN and their slaves, (4) IMM01 and their slaves, (5) CNGF and their slaves, and (6) the remaining programs. For each member P_j in a group, there is a point in a dot diagram which indicates $A(i, j)$. Thus, there are a total of 166 points in all density plots overlaid on a density plot in Figure 3 and Figure 4. In addition to the dot diagrams, we drew a vertical solid line at the mean average point $A(i)$.

Figure 3 shows the distributions of the average points in the average points profiles of DBSz, the three masters, and three slaves, one for each master. It is interesting to note that the average points of DBSz distribute unevenly; there are two distinctive clusters of points at around $x = 260$ and $x = 530$. The first cluster is created by the slaves, and the second clusters is created by TFTI's and other programs. The two clusters counterbalances each other, so that the mean average point of DBSz is near the middle point between two clusters (i.e. $x = 408$). We can reason about this phenomena in the following way: in most iterations in which DBSz plays with a slave, the payoff of DBSz is approximately equal to $1.3 = 260/200$, which is the payoff one may receive when mutual defection occurs in most of the 200 iterations in a game. In contrast, whenever DBSz plays with a program in the second cluster, it receives a payoff of $2.65 = 530/200$ in each iteration on average. In fact, a payoff of 2.65 is very close to what we expect DBSz can earn in a noisy environment. In the competition, the noise level is 0.1, thus about 20 iterations are affected by noise in a game. Suppose the mutual cooperation relationship has been established in a game and is never broken. Futhermore, suppose the other player is like TFT and it retaliates immediately when it receives a defection, but it would not remember this defection in future. Then DBSz receives approximately 20 defections in a game. Due to our policy of temporarily tolerance, we will cooperate in all iterations as long as the evidence collection process does not detect a change of hypothesized policy. Thus, we expect the point of DBSz of such a game is $540 = 3 \times 180 + 0 \times 20$ and the average payoff is 2.7.

DBSz also has a small cluster at $x = 580$. The cluster is formed by other DBS programs and some other programs. Once again it is close to our expectation: when DBSz played with another DBS, it received about 10 defections in a game, and the point of DBSz in this game is about $570 = 3 \times 190 + 0 \times 10$.

When compared with DBS's, the average points of the masters are distributed more evenly, even though they all have a cluster of points at around $x = 260$. Notice that in the distributions for

BWIN and IMM01 there are a number of average points that are over 600, and they are the average points earned by the master when they plays with their slaves. It indicates that BWIN and IMM01 can earn more points than what they can earn with mutual cooperation when they play with their slaves. Figure 3 also shows that the distributions for the slaves of IMS02 and CHHM are skewed toward to the lower end, but that for the slave of BWIN does not.

Another interesting thing about the distributions of average poinst for DBSz, lowESTFT_classic, TFTIm, Mod, TTFT, and mediumESTFT_classic as shown in the Figure 4 is that they all have two major clusters of points at around $x = 260$ and $x = 530$. DBSz has a shape cluster at $x = 530$, whereas other programs are more spreaded out. It seems there is a trend that the denser the clusters at $x = 530$, the higher the mean average points. Moreover, most of these selected programs except Mod cooperates well with DBS's and TFTI's; their average points against DBS's and TFTI's are often more than 500. Moreover, the the average points against DBS's form a small cluster near $x = 600$ in their density plots, and beyond $x = 600$ there is no more cluster. In other words, DBS's are among the most cooperative programs they encountered in the competition.

We notice that Mod does not work well with DBS's; the average point of Mod against DBS's often less than 500. We don't know why DBS's failed to cooperate with Mod. Our hypothesis is that DBS's sometimes do not cooperate with exploitative programs, which try to earn more points by defections. Mod's exploitative behavior can be seem in its density plot—some of its average points are beyond 600.

One might suspect that the clusters at $x = 530$ in many density plots are due to multiple copies of DBS's and TFTI's. To check whether it is the case, we removed all DBS's, TFTI's, and LSF, except DBSz and TFTIm, from the average point profile of DBSz, lowESTFT_classic, TFTIm, Mod, TTFT, and mediumESTFT_classic, and plot the graph in Figure 5. As can be seen, a dense cluster at $x = 530$ still exists in the density plot of DBSz, though it is somewhat less dense than that in Figure 4. It indicates DBSz is able to maintain a mutual cooperation relationship with many other programs submitted by other participants in the competition. Like DBSz, the clusters at $x = 530$ still exists in the density plots of other programs, but they become flatter after the removal. In general, the removal would lower the mean average points of these programs, but the lowering effect is greater for TFTIm, TTFT and mediumESTFT_classic than for DBSz and lowESTFT_classic, because their mean average points before the removal are smaller. The mean average point of Mod is less susceptible to the removal of DBS's and TFTI's. This result is consistent with the result in Figure 2.

To investigate the effect of slaves, we removed the average points due to slaves, in addition to our programs, from the average point profiles, and drew the density plots in Figure 6. This causes the clusters at $x = 260$ disappear or flatten in all density plots, and makes the clusters at $x = 530$ more prominent, especially the DBSz's cluster. Without the cluster at $x = 260$, the mean average points mainly depends on the center and the spread of the cluster at $x = 530$. Roughly speaking, the center of DBSz's cluster is the highest and the spread is the smallest among these programs.

1.5 A Comparison between DBSz and the Default Programs

We want to know why DBSz performs so well in the competition. However, as we know little about the strategies other participants used, our analyses of DBSz have to base entirely on the record of the moves other programs made in the competition. Due to a lack of information about the logics behind those moves, it is difficult for us to propose a theory to explain the performance

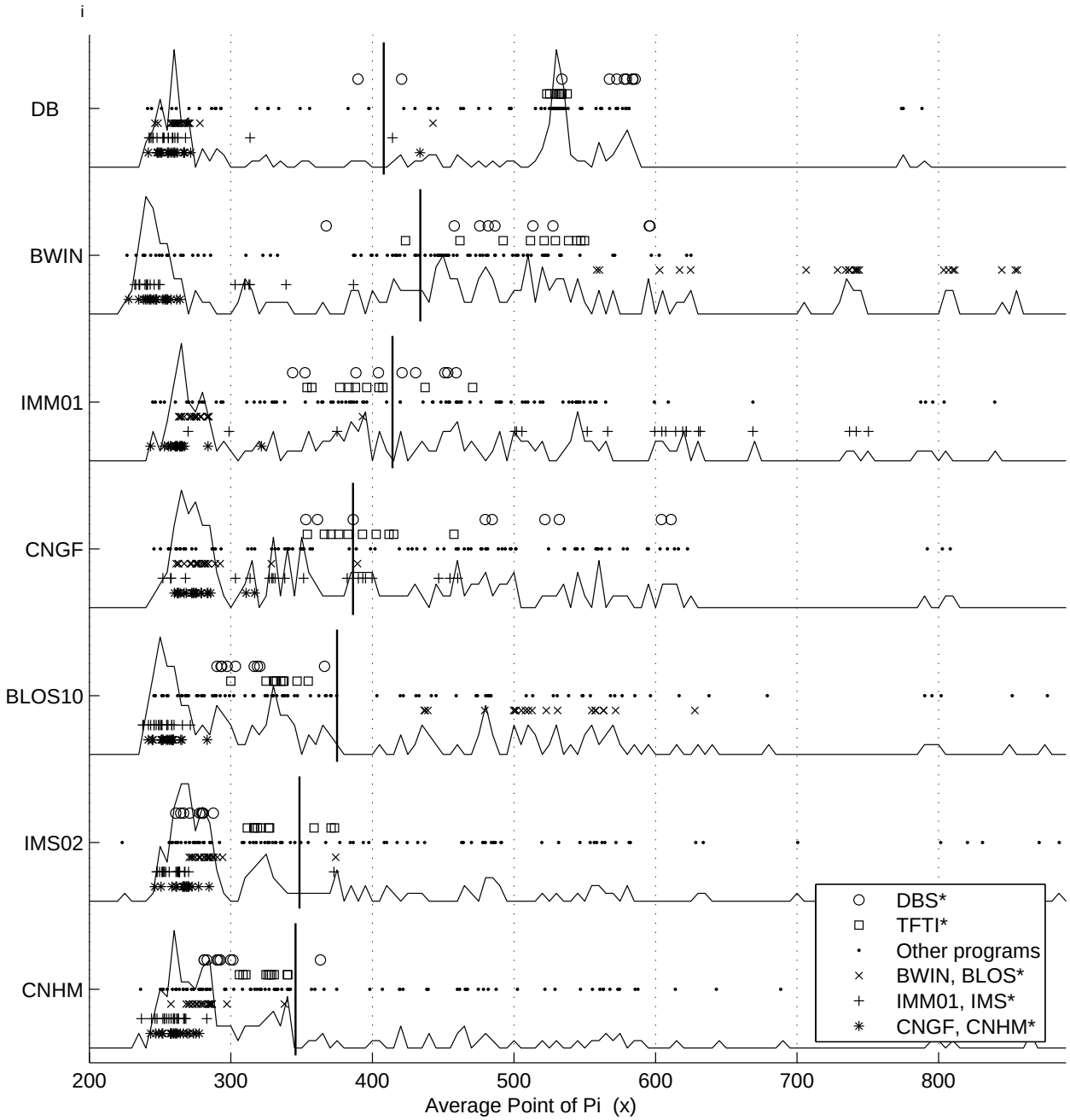

Figure 3: The distributions of average points in the average point profiles of DBSz and the master-slave strategies. The clusters of points at around $x = 260$ in all plots is due to the slaves. The BDSz's plot has a cluster of points at $x = 530$, which counterbalances the points at $x = 260$ (the mean average point is nearly halfway between two clusters). The average points of BWIN and IMM01 against some of their slaves are more than 600.

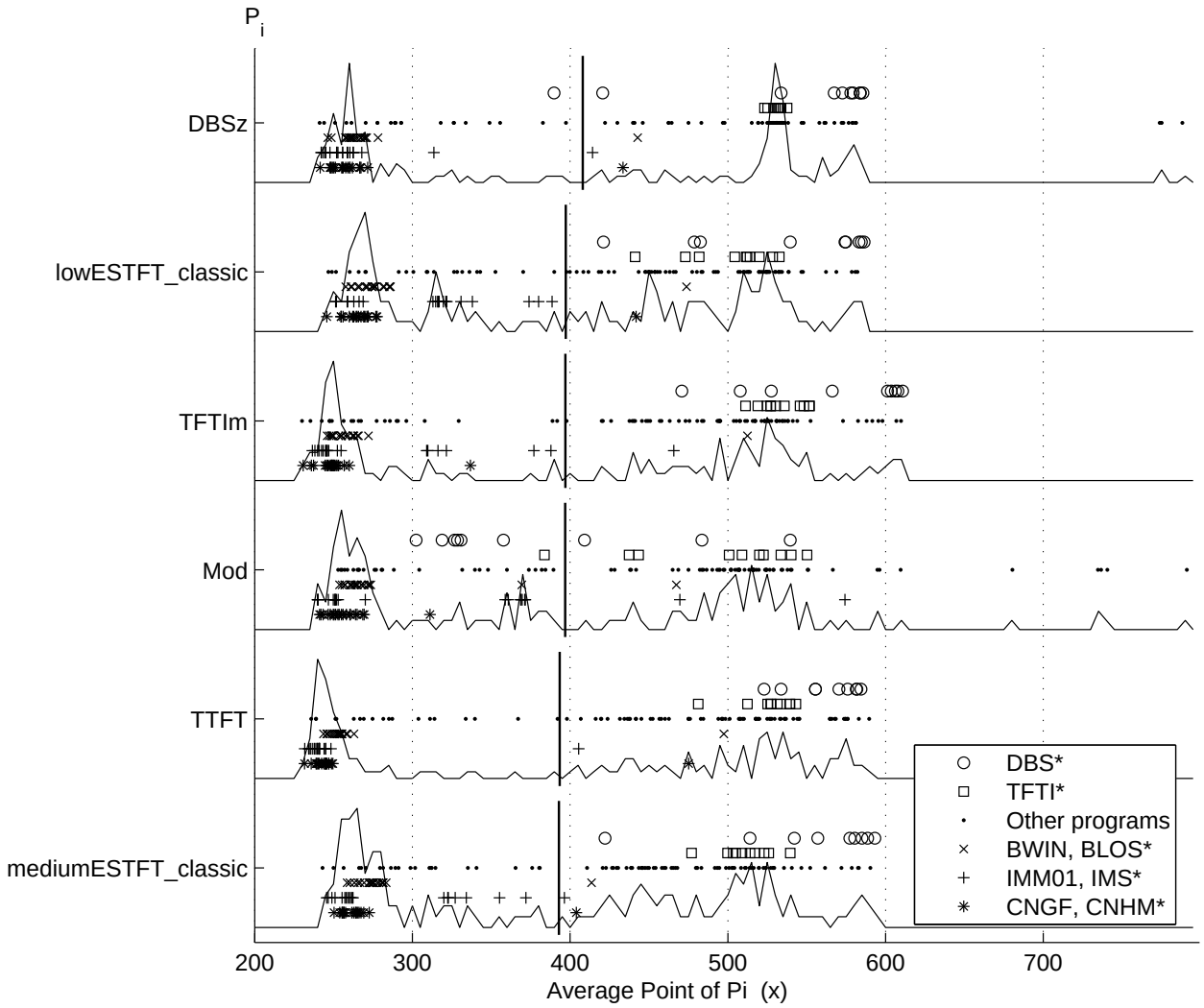

Figure 4: The distributions of average points in the average point profiles of DBSz, lowESTFT_classic, TFTIm, Mod, TTFT, and mediumESTFT_classic. The cluster of points at $x = 530$ is less spreaded out than the other programs'. All of these programs except Mod plays very well with DBS's and TFTI's; their average points against DBS's and TFTI's are often more than 500.

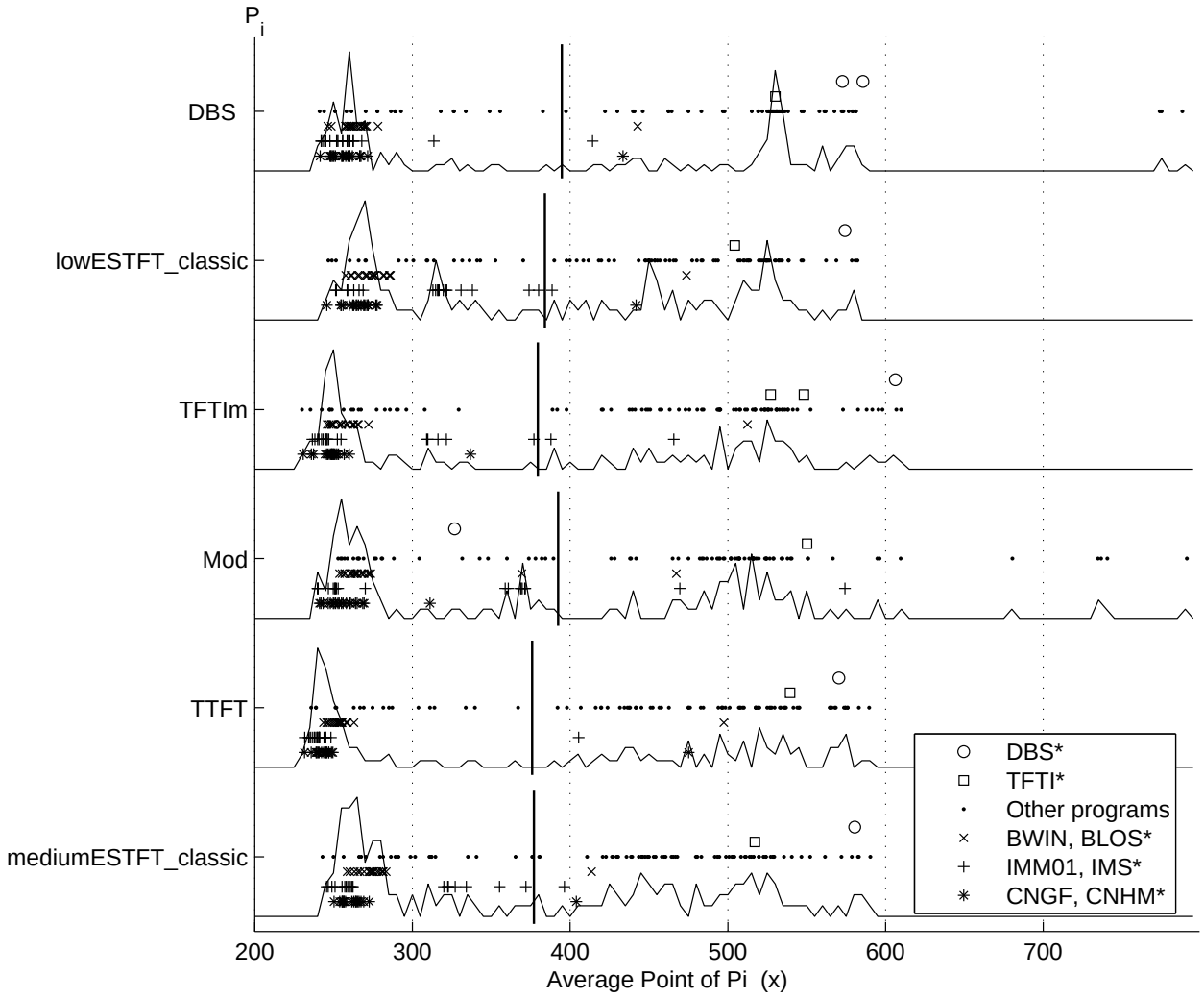

Figure 5: The distributions of average points in the average point profiles of DBSz, lowESTFT_classic, TFTIm, Mod, TTFT, and mediumESTFT_classic under the condition that the average points of all games against DBS, TFTI, and LSF (except DBSz and TFTIm) are excluded from their average point profiles. We can see that DBSz is still able to maintain a cluster at $x = 530$ without other DBS's and TFTI's. It indicates that DBSz cooperates with many other programs submitted by other participants. For other programs, the clusters at $x = 530$ become even more spreaded out when they are lack of supports from DBS's and TFTI's. The mean average points of all programs shift toward to the side where the clusters created by the slaves are located.

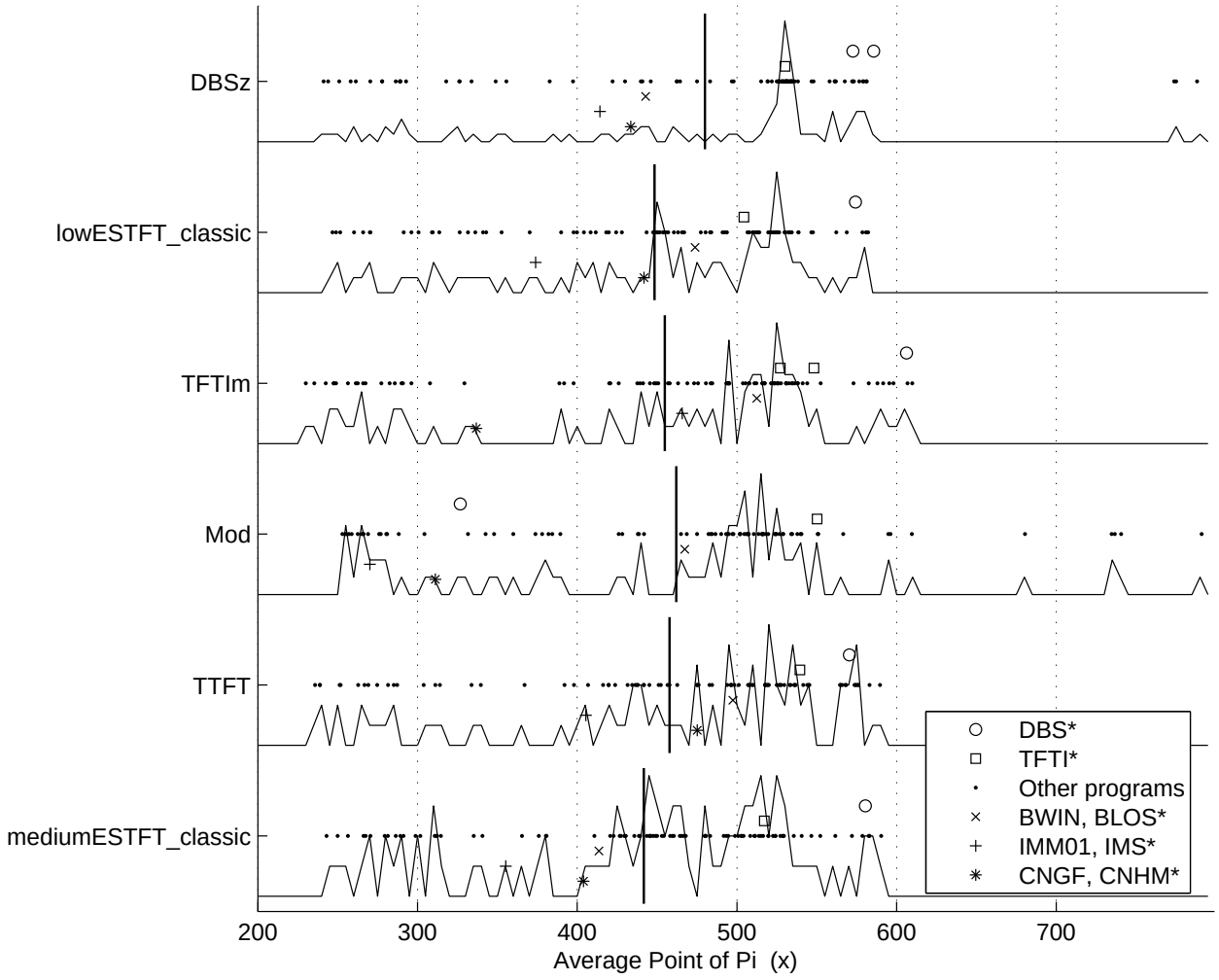

Figure 6: The distributions of average points in the average point profiles of DBSz, lowESTFT_classic, TFTIm, Mod, TTFT, and mediumESTFT_classic under the condition that the average points of all games against slaves, DBS, TFTI, and LSF (except DBSz and TFTIm) are excluded from their average point profiles. We can see that the clusters at $x = 260$ spread out or disappear in these plots. This makes the clusters at $x = 530$ more prominent.

of DBSz.

Fortunately, the organizer of the competition provided several simple but well-known strategies to compete with the programs submitted the participants in the competition. We call them the default programs. Since we all know how these default programs works, it is possible for us to explain why DBS performs better than them or vice versa. Hopefully, a study of this kind can shed light on the performance of DBSz when it encounters a player whose behavior is more or less similar to one of these default programs.

In the following, we shall focus on the performance of TFT, TFTT, GRIM, RAND, and DBSz in the games in which the other player is either DBSz or one of the default programs. A summary of their performance can be found in Figure 7.

Surprisingly, when compared with TFT, TFTT, GRIM and RAND, DBSz is the best program only when it plays with TFT, STFT, and ALLD; when it plays with other default programs, at least one default programs would outperform DBSz, and it is usually GRIM. There are two exceptions: when playing with TFTT, RAND is better than DBSz; when playing with DBSz itself, TFT is better than DBSz. Fortunately, the difference between the average point of DBSz and that of the best default program is not big.

In the competition, there are four rules in the hypothesized policy in DBS, and the conditions of these rules depends on the outcome of the previous iteration. It turns out when DBSz plays with a default program that the hypothesized policy is powerful enough to capture its behavior, DBSz usually performs very well. The simple strategy that can be completely modeled by the hypothesized policy are TFT, STFT, RAND, ALLC, ALLD, and NEG. DBSz got the highest points with playing with TFT, STFT, and ALLD when compared with other default programs in question. Even though DBSz loses to GRIM when playing with RAND and NEG, the difference between their average points are not big. However, problems occur only when DBSz plays with ALLC; most other default programs can outperforms DBSz, and DBSz earns much less than GRIM. It is because DBSz is not exploitative—it would not defect once it learn that it is playing with ALLC.

The reason why DBSz plays well with TFT and STFT is obvious—its initial hypothesized policy is TFT. With this correct hypothesized policy, DBSz always cooperates, which is the best response when playing with TFT. When playing with ALLD, the correct responses is always defect, and it will happen when DBSz learnt the behavior of ALLD. The reason why DBSz earns more than 200 points when playing will ALLD is the presence of noise; with 10% of moves are affected by noise, we expect DBSz earns slightly less than $230 = 180 \times 1 + 10 \times 5 + 10 \times 0$. But the reason why it is better than GRIM is unknown; the only explanation we can offer is that there are more noise in the games between DBSz and ALLD than in the games between GRIM and ALLD. The behavior of GRIM is similar to ALLD in noisy environments. It is why playing GRIM, programs earn as many as they earn when playing with ALLD. The best way to deal with RAND is to act like ALLD, but DBSz does so only after it rejects its initial policy. It is why its average point is slightly less than GRIM's when playing with RAND. The same reason can be used to explain the performance of DBSz when playing with NEG.

It is interesting to see that RAND outperforms DBSz when playing with TFTT. TFTT is known to do much better than TFT in noisy environment. In fact, it is true, as we see that the programs earn more points when playing with TFTT than with TFT in Figure 7. However, being more generous to defections of the other player, TFTT is susceptible to opportunistic defections by strategies such as RAND. Nonetheless, the difference of the average points between RAND and DBSz when playing with TFTT is not big.

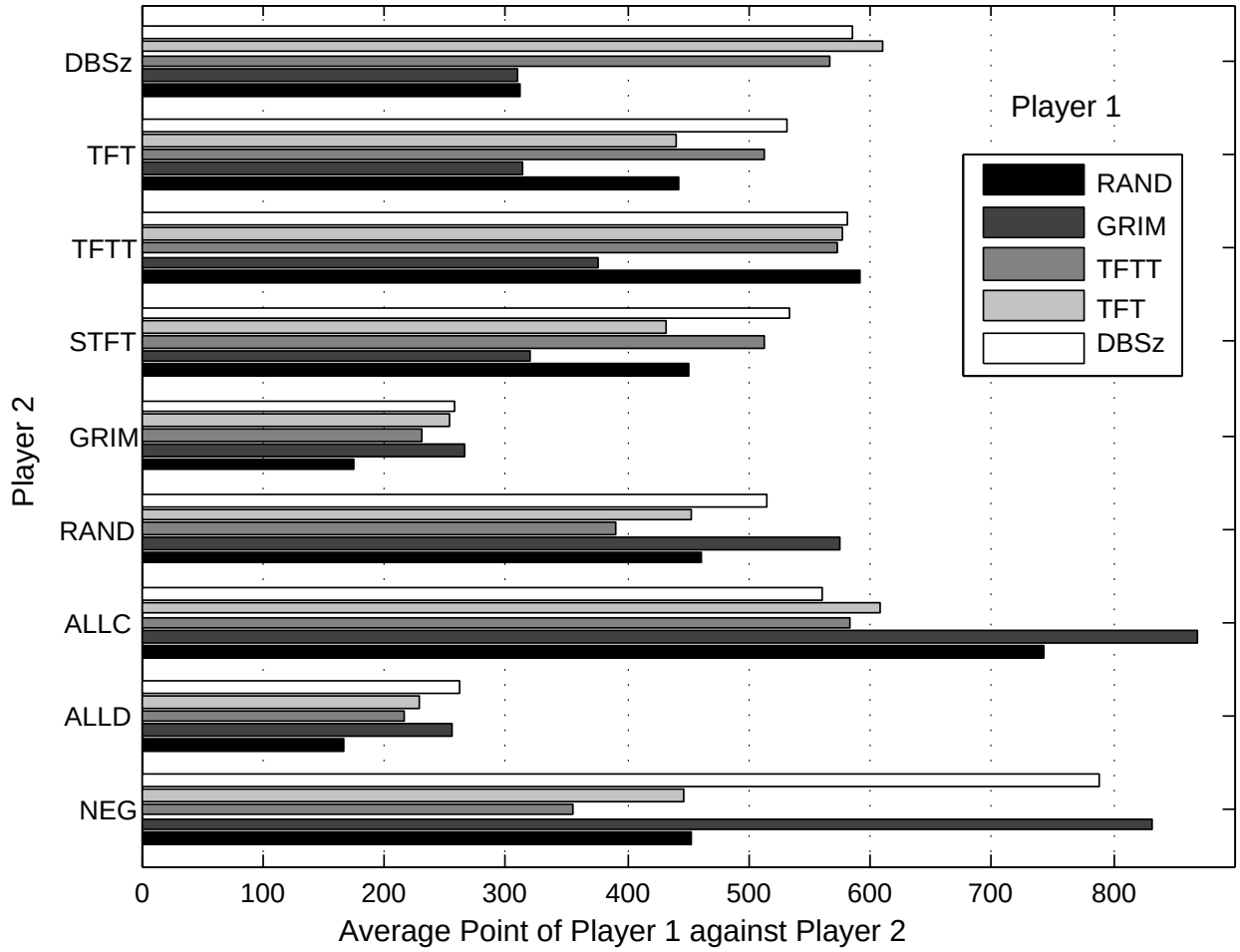

Figure 7: A comparison between the default programs (TFT, TFTT, STFT, GRIM, RAND, ALLC, ALLD, NEG) and DBSz. This figure shows the average point when player 1 plays with player 2, where the player 1 are chosen from the set of all default programs, while player 2 is either RAND, GRIM, TFTT, TFT, or DBSz.

References

[1] Graham Kendall. The iterated prisoner's dilemma competition: Celebrating the 20th anniversary. http://www.prisoners-dilemma.com, 2005.