Radboud University Nijmegen

Bachelor Thesis

Handing over control: When it is better for an artificial agent to let the human decide?

Douwe Heijligers

supervised by
Dr. Serge Thill

June 26, 2019

Abstract

This thesis considers human-robot collaborative scenarios, where the agent can be in need of assistance from a human collaborator. As for all autonomous systems, performance is very important. In addition to performance, it is also desirable that a human-agent collaborative system does not interrupt the human continuously: It needs to be able to pick good moments to do so, in order to not interrupt the human collaborator, while maintaining a good level of performance.

A collaborative chess game was implemented and used as an instrument to conduct experiments. In this game, two agents play against each other. One of these can decide to ask for help from an expert agent, based the distribution of evaluation values given a state. Using this framework, the research questions can be researched, pertaining to the level of performance of asking for help versus not doing so and whether the timing matters: We test this by comparing the performance of the former to an agent that hands over control at random intervals, but at the same frequency.

Experimental results showed that help-asking frequency influenced performance and that the timing at which these help-calls were uttered were important to the performance as well.

KEYWORDS: Human-Agent Collaboration, Human-Machine Interaction, Collaborative Systems, Game-playing agents, Uncertainty

Contents

1 Introduction

To tackle the problem of safe and dependable human-agent interaction, many different issues have yet to be solved. One of such issues pertains to a setting where human and agent collaborate to complete a common goal. In such a situation, it is important for the agent to know when it is in a state of uncertainty. This allows the agent to transfer control to the human agent, whom might be better in decision-making in a state of uncertainty.

Consider a scenario where an autonomous vehicle traversing traffic, does not know what the outcomes of its actions might be. With pedestrians around, and passengers aboard, it could be unethical to let the agent make an uninformed decision that could have consequences on the safety of humans. Instead, control could be handed over to the driver, whom might be better at judging which decision to make in this situation. Few research has been done on this topic, and most has been focused on when an agent can know when it can assist a human, but not the other way around. [1]

The questions that arise around this topic include *what* uncertainty is to an agent, which inherently entails how it can be formalized. Additionally, what threshold of this parameter is beneficial to the overall outcome of the human-agent collaborative system? With too much uncertainty, control will be handed over to the human without there being a need for it, in turn halting productiveness or efficiency. On the other hand, if an artificial agent is too 'certain', it might take unnecessary risks, which could have been prevented were a human involved in the decision-making process. In addition to the consequences on performance, the timing matters as well when the human is conducting an important task: It might not be desirable to interrupt a human who is carrying out an important task, if it is not absolutely necessary. There is a trade-off between improving the performance of the agent by asking for help, and interrupting frequency.

This thesis explores these research questions using a Chess game as an example, to simulate a degree of uncertainty. Chess is a game with an intractable number of possible outcomes, which encapsulates the meaning of the problem well. Because of the use of an evaluation function, pruning and tree search, a distribution of evaluation values over possible future states can be constructed. From this distribution, it is possible to formalize a measure of certainty an agent has in its belief that this choice will lead to a good outcome in the states after. Based on that information, a decision can be made to hand over control to a human, or expert agent. That can in turn be used to explore several interesting research questions, such as how performance changes as help asking frequency changes and what the timing of asking contributes to this performance change.

Using a simulated environment such as the game of Chess, facilitates constraining the research and makes learning about the agents behaviour more

straightforward, as physical aspects and perception components of intelligence can be ignored [2], allowing more focus on the decision part of the problem, and in turn, the effect of handing over control to a human.

The focus of the experiments pertains to exploring the following questions:

In a situation where an agent collaborates with an expert agent to achieve a shared goal,

1. Does the frequency of asking an expert agent for help when in a state of uncertainty aid in increasing performance compared to a scenario where this is not the case?

2. Is the increase in performance merely due to the frequency, or is the timing an important factor as well to the overall performance of the agent-agent collaborative system?

When the latter question can be answered with yes, this could also mean that the formalization of uncertainty used for the collaborative agent is useful and informative, leading to a certain increase in performance.

2 Background

2.1 Cooperative Agents

Human-agent collaboration is a field of research with a very broad scope, and every day more technologies become available that allow us to move from traditional applications of robots to an integration of them into more complex human work environments. [3]

Cooperative Human-agent interaction techniques are already being applied in novel and increasingly complex domains such as elderly care [4] or rescue operations [5]. When two or more participants share a common intention, they can communicate with each other to share their intentions and action plans. A good example of such a system is an autonomous transport solution such as a self-driving cars. In such a situation, the human (driver) and agent (software and sensors) perform as a *collaborative mixed initiative system*, which in principle will yield to significant benefits considering the performance of the overall system [6].

Trust of humans in automated solutions is of critical importance to the performance of a collaborative system. That is, these systems will only be able to truly succeed if the human has the appropriate level of trust in the artificial agent's abilities [6].

2.2 Applications

The field of application for human-agent collaboration systems is very diverse. These various diverse fields, such as construction, healthcare, transport, or search and rescue, all have very different requirements, problems, and levels of integration of the agent which become evident when considering the amount of interaction between the human and the artificial agent that is necessary for the joint application to function successfully. It should be mentioned however, that all domains share one important aspect, which is that the artificial agent has to perform in an uncertain environment.

A good example of a domain where artificial agents are starting to take on a prominent role as part of a human team, is **search and rescue**. Robots can be small, and get in to places where humans can not. They can collect important data using various sensors, and assist search and rescue teams by being able to go places that are to dangerous, or hard to reach. One of the first situations that arose where the sole purpose of an artificial agent is not to *replace* a human, but to assist it, in rescue operations, were the attacks on the World Trade Center during 9/11. In these circumstances, the helper robots could provide important additional features that make the work of the rescue team safer, more dependable and more efficient. [7]

Another example worth mentioning is in the field of **construction**. Robot assisted task-design is useful here because the helping agent can provide relief from lifting heavy loads, providing dependable and precise manipulations of materials, and making the repeating nature of tasks less cumbersome. A robot that can do so is the Mobile Robot Helper by Kosuge et al. [8], which can cooperate with a human to lift heavy objects.

Yet another good example of domain where HRI can play a role is **entertainment**. In this field of interest, various companion robots exist. Examples of such applications include dogs such as Sony's AIBO, which can communicate with humans as well as other agents. Artificial agents such as these could comfort and entertain people [9].

These companion robots can also be linked to the aforementioned application domain of **healthcare**. As an example, the robot *Paro* which has been shown to be able to relieve feelings of loneliness in elderly [10]. According to Robinson et al., experimental results showed that when compared with the control group, residents of an elderly home who interacted with the companion robot had significant decreases in feelings of loneliness.

Most topics discussed above are still heavily researched, however, as time progresses, and research advances, such systems will become more commercially available and dependable as performance increases, as Bauer et al. state [3]. Additionally, Bauer makes the point that the most important focus of design

for any human-agent collaborative system should be on **safety**.

These topics extend well to the subject of this thesis because handing over control to a human in an uncertain situation could and should be beneficial to the safety of humans. Not only that, it could increase performance, efficiency and durability of workforce by assisting humans with physically demanding work, or providing sensory capabilities that humans would otherwise miss. Agents might also be able to assist humans in other ways. Exploring what changes about the performance of an agent when deciding for help, is therefore an interesting topic for this thesis.

2.3 Measures of performance for collaborative systems

The fast growing research field of cognitive robotics and Human-Machine interaction has seen many developments over the last years. However, one major challenge still remains for systems in which humans and artificial agents collaborate: How one defines clear metrics and benchmarks to measure the performance of such a system.

Aly et al. provide a useful framework for the evaluation of an agent in a human-machine or human-agent interaction scenario. [11] Important factors to take into consideration when evaluating the effectiveness of an agent in such a scenario include: persuasiveness, awareness of interaction and engagement. [6]. Persuasiveness and engagement could be important because they might influence the likelihood of the human accepting the decisions, and proposals, made by the artificial agent. If the agent scores low on these measures, it might have implications on the overall performance of the human-agent collaborative system.

2.4 Human-Robot Trust

An important factor in the performance, user experience and efficiency to interact with a robot in a collaboration task is trust in the robot. [6] The most important predictor for measuring a great degree of trust of a human in a robot is performance. Environmental factors are also associated, but human factors are less important [12].

Trust is important in a collaborative HRI scenario, because it determines how likely a person is to accept the information that is produced by the agent, or follow directions or suggestions given by the agent. This especially holds for risky and uncertain situations [6] [13]. This implies that a human will not accept proposals made by the agent, or will continue on its own, when it does not trust the agent. This might be dependant on the help-asking frequency: Asking for help too often, or interrupting at bad timings, might have consequences on the collaborative behaviour of the human, stunting performance, thereby having implications on dependability, efficiency and perhaps safety.

3 Methodology

3.1 Research questions and hypotheses

As stated in the introduction, this thesis explores the following two research questions and their respective hypotheses:

In a situation where an agent collaborates with an expert agent to achieve a shared goal,

1. Does the frequency of asking an expert agent for help when in a state of uncertainty aid in increasing performance compared to a scenario where this is not the case?

   - H_0 : Performance does not change with higher frequencies of asking an expert agent for help.
   - H_1 : Performance changes when asking an expert agent for help more often.

   Building on the results we obtain from this research question, the following question can be asked:

2. Is the increase in performance merely due to the frequency, or is the timing an important factor as well to the overall performance of the agent-agent collaborative system?

   - H_0: An agent which asks at random intervals (uninformed) but at the same frequency performs equal to an agent that bases its decision on the distribution of evaluation values.
   - H_1: An agent which asks at random intervals (uninformed) but at the same frequency performs differently when compared to an agent that bases its decision on the distribution of evaluation values.

Exploring the first question could be useful because as far as asking for help goes, one wants the frequency to not be too high, as to not interrupt the human too often. It is interesting to see whether asking for help at a reasonably low frequency is still significantly better performance-wise than not asking for help. Additionally it is interesting to see how performance changes for different frequencies.

The second question is interesting because it shows whether the measure implemented to flag a state as uncertain and basing a decision to hand over control on that state, actually has any value: Is it significantly better than asking at random intervals?

Two types of agents play against each other to explore these research questions:

- `SmartAgent`: This type of agent uses MiniMax with alpha-beta pruning to decide what move to pick.

- `CooperativeAgent`: This type of agent uses Minimax with alpha-beta pruning as well, but has the additional functionality of being able to hand over control to another (better) agent, or a human, given a state of uncertainty. The decision to hand over control can either be informed (based on a threshold over the distribution of evaluation values), or uninformed (ask at a certain frequency, but at random times).

To answer above questions, a series of experiments is conducted where a `SmartAgent` plays against a `CooperativeAgent` on various settings, with varying search depths, threshold values, and informed/random help-asking conditions. More technical details about the types of agents and how they are implemented can be found in section 3.2.

3.2 Implementation and algorithms

The Chess game has been implemented in Java, based on Chess logic from a framework found on `github`: (`https://github.com/gsig/java-chess-ai`, GNU General Public License v3.0). A graphical user interface was built around this framework and the Minimax algorithm with alpha-beta pruning was modified, which forms the foundation of the decision making for both the fully autonomous- and the collaborative agent.

3.2.1 Implementation

The program consists of four panels:

1. The board, where the game of Chess is played. A player can click on a tile containing a piece and after on another tile to move the piece to the respective location, provided that the cooperative agent has asked for help.

2. The information panel, containing textual status updates from the cooperative agent.

3. The statistics panel, containing various metrics and parameters showing the current state and configuration of the game.

4. The metrics panel. On this panel, a small chessboard is drawn, and for every possible move that the cooperative agent can make given it's current state, an evaluation value is calculated and scaled by color on the small chess board. Using this feature one can observe, through the presence or absence of color differences, what the distribution of evaluation values over all possible moves is. Either as an informative statistic to provide clarity as to which states the agent is doubtful about, or as a guide to provide insight in what move to pick should the agent have asked for help. It is possible to click on a tile in the metrics panel to view the underlying moves and respective evaluation values that lead to this tile.

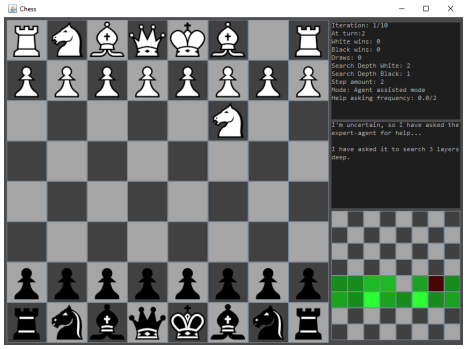

Figure 1: Layout of the Chess playing program.

3.2.2 Agent algorithm

The Chess-playing algorithm works as follows:

1. Upon initialization, a value is assigned to every piece, which is provided by the chess-programming community (`https://medium.freecodecamp.org/simple-chess-ai-step-by-step-1d55a9266977`). Using this table, different pieces are assigned different evaluation values, since some pieces have more degrees of freedom, or play a more important role and are therefore more valuable than others. This evaluation table can be seen in figure 2.

Figure 2: The values for each piece on the board.

2. In addition to this inherent value each piece has, a piece also has a two-dimensional matrix that contains evaluation values. The values in this matrix correspond to the desirability of a certain position to a piece. As an example, a Knight is most useful when located in the center of the board, for at that location, it has the most possible moves it can make. Located at the side, it loses two degrees of freedom, so the matrix reflects that by assigning a negative value to it. An example as provided by the chess-programming community can be seen in figure 3.

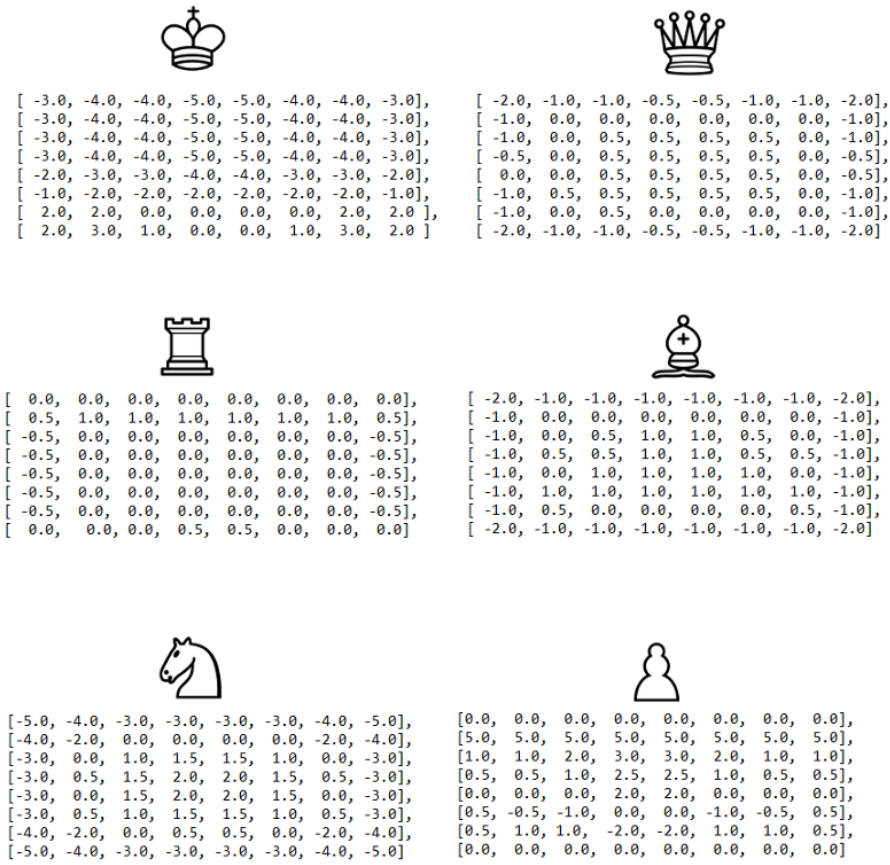

Figure 3: Board position evaluation values per piece.

3. Using this table, we can construct a search tree using Minimax. Any two player game that contains a competitive element is likely to have the property that a move that is beneficial for one player, is detrimental to the other. This is the foundation of Minimax, while one player tries to maximize its evaluation, while the other tries to minimize it, leading to a sum of 0 over both evaluations. A visual example of Minimax can be seen in Figure 4.

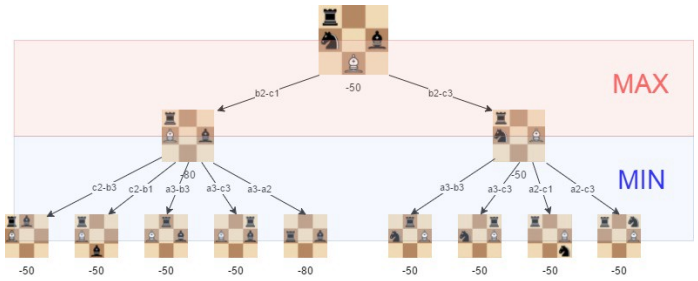

Figure 4: Minimax visualization.

Simplified pseudocode for a Minimax algorithm:

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := -INFINITY
        for each child of node do
            value := max(value, minimax(child, depth  1, FALSE))
        return value
    else (* minimizing player *)
        value := +INFINITY
        for each child of node do
            value := min(value, minimax(child, depth  1, TRUE))
        return value
```

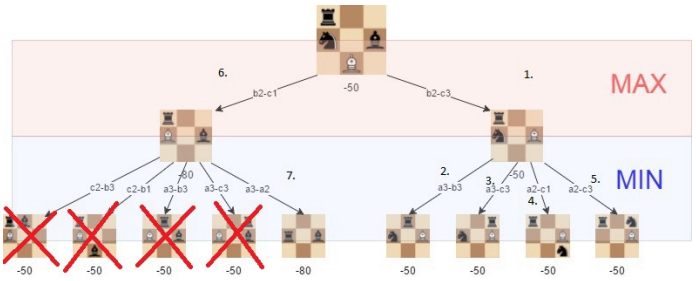

Figure 5: Minimax visualization with Alpha-Beta pruning

4. Implementation of Alpha-Beta pruning is necessary for Chess, because the number of evaluated moves is arbitrarily large, quickly becoming intractable. The pruning relies on the situation where we can stop evaluating a part of the search tree if we find a move that leads to a worse situation than a previously discovered move, as can be seen in figure 5.

5. Above mentioned techniques are the ones used for the `SmartAgent`, the type of agent that relies on computation only to make its decisions. This will be the agent on the other side, against which we will test our `CooperativeAgent`. For this collaborative agent however, it is necessary, that additional features are added, in order to be able to hand over control to the human player, or expert agent.

3.2.3 Formalization of uncertainty

The `CooperativeAgent` needs to construct a distribution of utility values for every legal move given a board state. Therefore, for every legal and directly executable move, a search tree is opened (up to a certain search depth) and moves with highest evaluation values that can follow from that tile are added to a list. Given this list, the frequencies of evaluation values can be calculated. Should the distribution be heavily peaked around its highest evaluation value, then it might be that the agent is in an uncertain state.

An example of a situation that could be uncertain is the opening state seen in figure 1. Black can choose out of 20 possible moves. Four of these have a significantly larger evaluation value than the others (Moves where the Knight moves towards the center of the board, brightest green, or, leading to the same tile, moves where respective Pawns move to the same tile). It could be said that such a situation might be uncertain, while both moves are equally good according to the evaluation function.

Determining when an agent is in a state of uncertainty plays an important role. Therefore, a formalization for this concept is necessary. In order to determine a state of uncertainty, the following procedure is used, where we can vary

the threshold t at which it flags a state as uncertain for use in our experiments with human-robot collaboration.

- First the returned list of evaluation values E has to be normalized, on the interval $\{0, 1\}$.

  - Given a list of positive and negative reals $E \in \mathbb{R}$:
  - Shift to positive by adding the minimal value of E to all elements of

$$E : \forall e \in E, e \leftarrow e + min(E)$$

    ,
  - Normalize with the following formula:

$$\forall e \in E, e \leftarrow \frac{e - min(E)}{max(E) - min(E)}.$$

- Next, E is transformed to a set $S = \{e, f\}$ of unique evaluation values and respective frequencies, obtaining the frequency of the maximal value f_{max}, an uncertain state u may be flagged using the following function:

$$u(t, f_{max}) = \begin{cases} 1, & \text{if } f_{max} \geq t \\ 0, & \text{otherwise} \end{cases}$$

  Where 1 corresponds to an uncertain state. In other words, the algorithm flags a state as uncertain when it encounters a frequency of the maximal evaluation value $f_{max} \geq t$ given a board state.

- t may be a set value, for example, if $t = 2$, then when there are two or more states with maximal evaluation value, control will be handed over. However, a function can also be assigned to t, such that the frequency is compared to a fraction of the moves in the considered set, for example $(t = \#moves/value)$. This is useful as a middle ground, because the number of moves one can make usually varies between 18 and 30 from early game to pre-endgame. This therefore yields a more dynamic threshold that varies between approximately 2 and 4 when $value = 8$. Note that if $t = 1$, than the agent will always ask for help, while there is always at least one maximal move for every game state. Another function that is used for t takes the list of moves, transforms it into a dictionary, takes the two highest evaluated moves, and adds there frequency. That function is referred to in this thesis as $moves2$. As can be expected, when $t = moves2/value$, the number of times asked for help is approximately 2 times the size of $moves/value$.

3.3 Experimental setup

The series of experiments used to explore the topics of this thesis have in common that a `SmartAgent` plays against a `CooperativeAgent`, in all but the benchmarking scenario, which is used to explore what higher search depth does to the search depth, as it increases time per simulation tremendously. Therefore if it is possible to obtain similar results with lower search depths, that allows for running more simulations per experiment.

The obtained performance data consists of the number of wins per color for each trial, the number of draws, the tournament score (explained below) for each color, the proportion of score obtained per color and, for the collaborative agent, the frequency of asking for help per turn.

3.3.1 Performance evaluation

For evaluating performance in different scenarios, two types of agents will be used, namely the SmartAgent and `CooperativeAgent`, as mentioned before. The following parameters are adjusted for different scenarios:

- Search depth = sd,
- Expert agent search depth sd_e,
- Number of trials = N,
- Uncertainty threshold t.

For performance evaluation, the following metrics are used:

- The number of black wins,
- The number of white wins,
- The number of draws,
- Frequency of help-asking per turn f,
- Tournament score for black $s_b = \#wins * 3 + \#draws$,
- Tournament score for white $s_w = \#wins * 3 + \#draws$,
- Proportion of score obtained by black: $p_b = \frac{s_b}{s_b + s_w}$

The tournament score is built up as follows: A win yields 3 points, a draw yields 1 point, and a loss yields 0 points. Traditional Chess scoring systems use the system that rewards a win with 1 point, and a draw with $\frac{1}{2}$ point [14]. In Chess tournaments, the 3-1-0 scoring system is somewhat experimental: It has been used in some tournaments over the past decade, such as the Bilbao Masters Tournament [15]. Because it rewards wins more generously than draws, it is perhaps better suited for these experiments. The reason being that some

experiments may reach a deadlock state (both sides oscillate, and the inherent randomness of the algorithm can not get them out of this oscillation). In such a state, the cutoff for a game lies at 200 turns and will always yield a draw. Even though these occur quite rarely, they should not yield a large fraction of points, which led to the decision to use the 3-1-0 scoring system. Additionally, using a scoring system like this as a metric makes draws more attractive to black than losing, as opposed to when one just uses the number of wins for black. This way, scenarios that yield more draws are still preferred over ones where white gets to win. In all scenarios, the `CooperativeAgent` plays as black.

- **Benchmark**
  For a benchmark of the `SmartAgent`, the agent plays against itself, on equal and different search depth settings. Expected is to see that the games should be reasonably uniformly distributed considering wins between agent A and B, when their search depths are equal.

  1. `SmartAgent` A ($sd = 2$), `SmartAgent` B ($sd = 2$), $N = 200$.
  2. `SmartAgent` A ($sd = 2$), `SmartAgent` B ($sd = 3$), $N = 200$.
  3. `SmartAgent` A ($sd = 3$), `SmartAgent` B ($sd = 3$), $N = 200$.

- **Determining search depth settings**
  The benchmark experiments are followed up by a series of experiments of games where `SmartAgent` plays against `CooperativeAgent`, but at different search depth settings. $t = moves/8$ for all three experiments, where *moves* is a function returning the number of moves that have maximal evaluation value given a state.

  1. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $N = 20$.)
  2. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 1$, $sd_e = 3$, $N = 20$.)
  3. `SmartAgent` ($sd = 3$), `CooperativeAgent` ($sd = 2$, $sd_e = 4$, $N = 20$.)

- **Frequencies for different thresholds**
  These are followed up by a series of experiments of agent-agent games, to collect data about the performance of a collaborative system for different thresholds of uncertainty while asking for help. In these experiments, the threshold for uncertainty varies from low, $t = 1$, always asking for help, to high ($t = \infty$), never asking for help. *moves* refers to the number of moves that have the maximal evaluation value given a state. $moves2$ refers to a function that returns the number of moves that have maximal evaluation added to the number of moves with 2^{nd} best maximal evaluation.

  1. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = 1$, $N = 20$.)
  2. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = (moves2/5)$, $N = 20$.)
  3. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = 2$, $N = 20$.)

4. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = moves/8$, $N = 20$.)
5. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = moves/6$, $N = 20$.)
6. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = moves/4$, $N = 20$.)
7. `SmartAgent` ($sd = 2$), `CooperativeAgent` ($sd = 2$, $sd_e = 3$, $t = \infty$, $N = 20$.)

- **Performance vs. help-asking frequency**
  From above experiment average frequencies of asking for help per turn are obtained. From these, performance scores are plotted, and a line is fitted to the histogram to obtain a function of performance against help-asking frequency per turn, for both game outcome (black win/white win/draw), as for the score of `CooperativeAgent`. The first hypothesis can be rejected or accepted based on the obtained regression statistics for the score of black, and the number of wins for black.

- **Informed timing vs. random interval**
  After above experiment, scenarios at non trivial frequencies (not 0 or 1) will be tested with an uninformed, adapted `CooperativeAgent` that asks for help at random, however, at the same frequency as the informed agent. This agent will play against the `SmartAgent`, under otherwise identical conditions as above experiment.

  After data has been gathered, we can compare the slopes of the two regression lines, and look at the regression statistics. Both as score of `CooperativeAgent`, as the proportion of the total score `CooperativeAgent` has accumulated for each scenario.

4 Results

In all experiments, a `SmartAgent` plays white, while the agent we are experimenting with (mostly `CooperativeAgent`), plays black.

4.1 Game performance

4.1.1 Benchmarks and determining search depth settings

The following results were obtained when running `SmartAgent` versus another `SmartAgent`, $N = 200$, at different levels of search depth sd:

Configuration	White wins	Black wins	Draws
$sd1$ vs $sd1$	83	87	30
$sd2$ vs $sd2$	50	49	101
$sd3$ vs $sd3$	48	53	99

Table 1: Results of the first benchmark test.

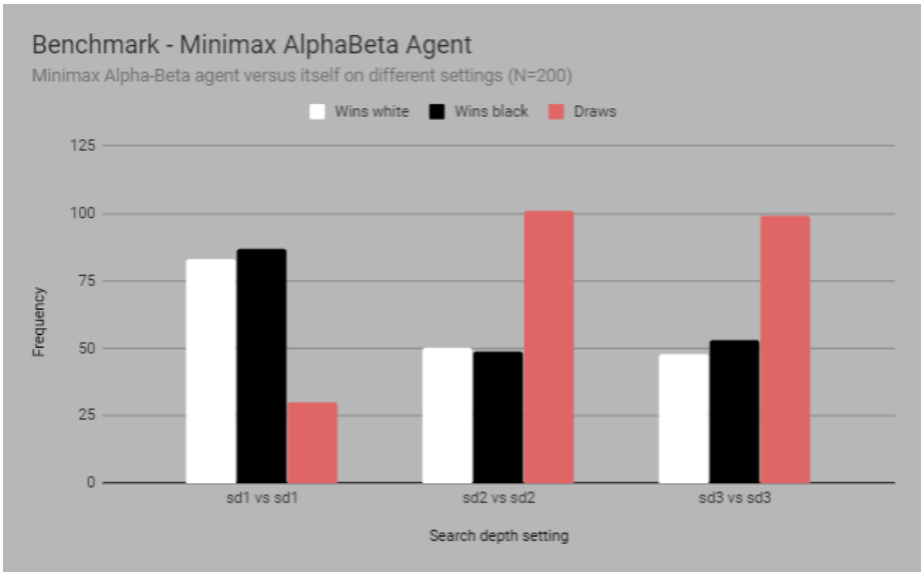

Figure 6: Benchmark 1: `SmartAgent` vs. `SmartAgent`.

As can be seen from figure 1, the number of wins that black has are more or less equal to those of white for all tested search depths. The number of draws increases dramatically as the search depth increases from 1 to 2.

The next simulation ran a `SmartAgent` against a `CooperativeAgent`. From these the goal was to select a scenario in which the `CooperativeAgent` outperformed the `SmartAgent` by a good margin, while keeping the search depth as low as possible (to reduce unnecessary runtime). These resulted in the following data ($N = 20$):

Configuration	t	Wins white	Wins black	Draws	Score s_b
$sd3$ vs $sd2, sd_e4$	moves/8	5	9	6	30
$sd2$ vs $sd1, sd_e3$	moves/8	8	10	2	31
$sd2$ vs $sd2, sd_e3$	moves/8	2	10	8	34

Table 2: Results of the second benchmark test.

From the table, as well as figure 7, it is visible that a good scenario is the last, $sd2$ vs $sd2, sd_e3$, because the tournament score for black s_b is highest, while computationally it is less expensive than the first scenario, with better results. Compared to the second scenario, it is better because it is only marginally slower, yet yields much less wins for white.

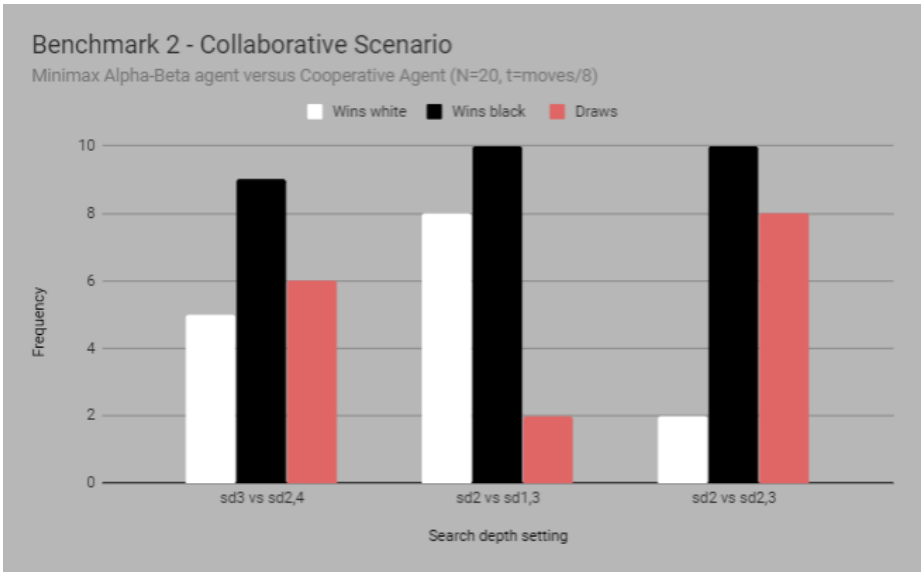

Figure 7: Benchmark 2: `SmartAgent` vs `CooperativeAgent`.

4.1.2 Frequencies for different thresholds

The obtained average help-asking frequency per turn f from the simulations on $sd2, sd2, sd_e3$,
$(N = 20)$:

Threshold t	f_{approx}	f_{exact}
$t = 1$	1	1
$t = moves2/5$	0.5	0.4796
$t = 2$	0.35	0.3468
$t = moves/8$	0.25	0.2625
$t = moves/6$	0.2	0.2023
$t = moves/4$	0.125	0.1254
$t = \infty$	0	0

Table 3: Obtained frequencies for different threshold settings.

The exact frequencies are rounded to the nearest fraction for clarity. As can

be observed from the data, the frequency goes down as the number of moves that has to be in the considered set goes up. When this threshold is 1, the agent will always ask for help (while it is always *at least* considering 1 move). When t is very high, it will never ask for help, because there will never be enough moves in its considered set. When a fraction of $\frac{1}{5}$ of the output of function $moves2$ is used, we obtain a state of uncertainty in approximately half of the game states.

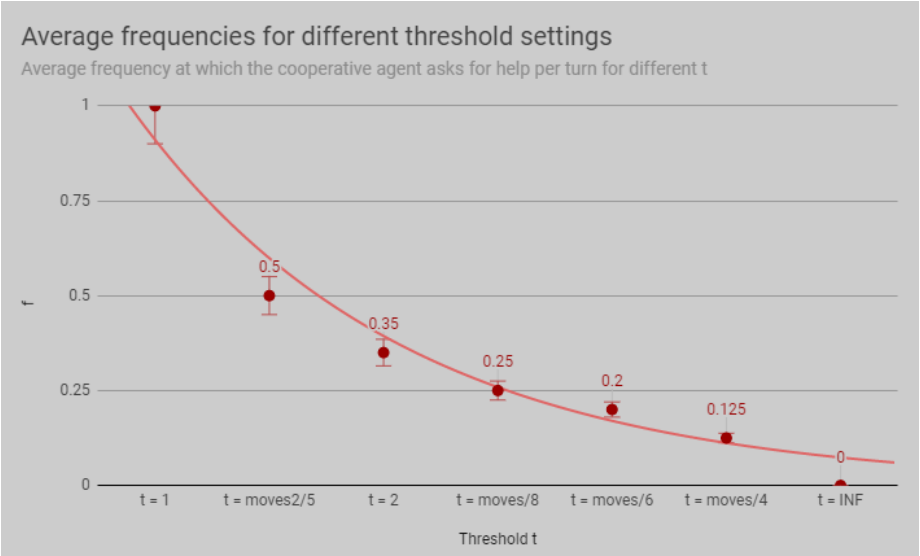

Figure 8: Frequencies for different thresholds.

4.1.3 Performance vs. help-asking frequency

After obtaining a good configuration to test on, as well as finding the frequencies for several thresholds on this configuration, experiments on the performance of the agent have been conducted for all these frequencies on aforementioned configuration of $sd2$ vs. $sd2, sd_e3$. The following data was obtained:

Frequency f	Wins black	Draws	Wins white	Score black s_b
1	16	4	0	52
0.5	10	7	3	37
0.35	11	7	2	40
0.25	10	8	2	38
0.2	9	7	4	34
0.125	4	12	4	24
0	4	10	6	22

Table 4: Data of performance by frequency.

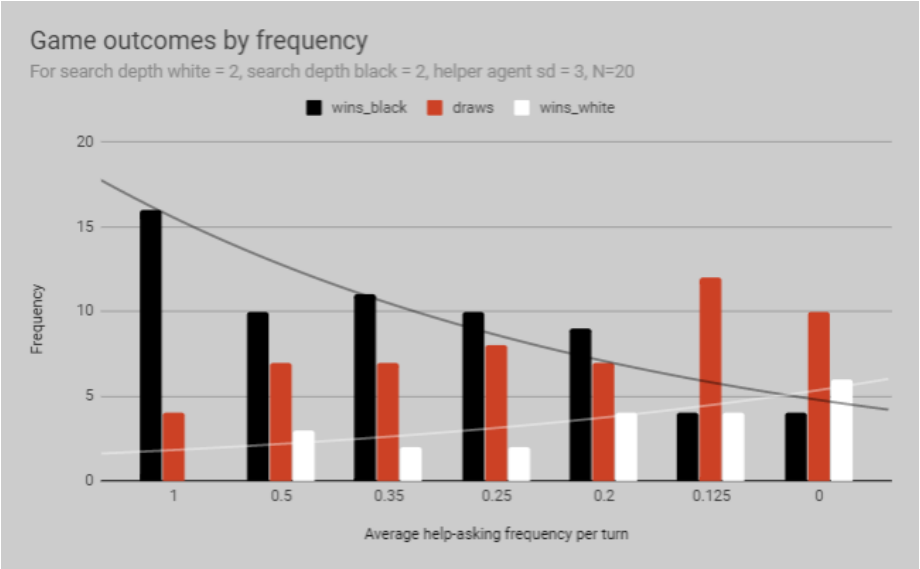

Figure 9: Game outcomes for different frequencies.

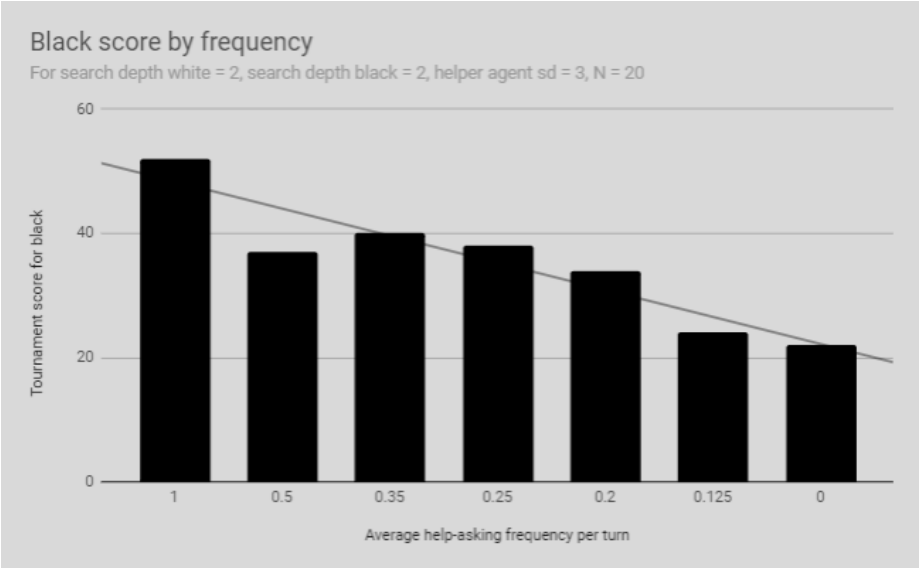

Figure 10: Tournament scores for black s_b for different frequencies.

Given obtained data, a linear regression was conducted, to see if the decrease

in performance was significantly explained by the decrease in frequency. By looking at the slope of the regression line, it is possible to conclude whether it is significantly non-zero.

- For the number of wins for black, there was a **significant** effect of frequency f on the number of wins for black:
  - The coefficient of the fitted line is 11.50, with an error of 2.49.
  - The 95% HDI lies between 5.301 and 17.69.
  - This slope is significantly non-zero with $p = 0.0050 \leq 0.05$, $R^2 = 0.8199$.
- For black's tournament score s_b, there was a **significant** effect of frequency f on s_b:
  - The coefficient of the fitted line is 28.03, with an error of 5.72.
  - The 95% HDI lies between 13.32 and 42.73.
  - This slope is significantly non-zero with $p = 0.0045 \leq 0.05$, $R^2 = 0.8276$.

Given that frequency has a significant non-zero slope with both the wins for black, and its respective tournament score (factoring in draws favored over losses), which are both performance measuring metrics, it can be concluded that a decrease in frequency of asking for help when in a state of uncertainty has a significant (negative) effect on the performance of the `CooperativeAgent`.

Hypothesis: H_0 can be **rejected**, as the frequency of asking for help has a significant effect on both score and number of wins, and therefore the performance of the collaborative system.

4.1.4 Informed timing vs. random interval

The second research question pertains to whether the timing of asking for help is important. Therefore another set of runs is executed, with the difference that the threshold of asking for help is not longer included in determining when to ask for help. For these experiments, $f = 0.125$ has been omitted, while at such a low help-asking frequency, there is no longer a significant advantage to ask for help, as can be seen from figure 10.

Instead, the agent asks for help based on a random function, sampling a value between 0 and 100, and the threshold is placed somewhere in this interval (e.g. for a help-asking frequency per turn of 0.5, this threshold is set to 50, when the random number generator returns a value equal or greater than 50, the agent asks for help).

These results are then compared to those of the informed condition, where the

decision to hand over control is based on the frequency of maximally evaluated moves. The following data has been obtained ($N = 20$):

	Informed			Random		
f	s_b	s_w	p_b	s_b	s_w	p_b
0.5	37	16	0.6981132075	44	14	0.7586206897
0.35	40	13	0.7547169811	33	21	0.6111111111
0.25	38	14	0.7307692308	32	20	0.6153846154
0.2	34	19	0.641509434	27	24	0.5294117647
0.125	24	24	0.5	30	27	0.5263157895

Table 5: Score and proportion data for informed vs. random timing conditions.

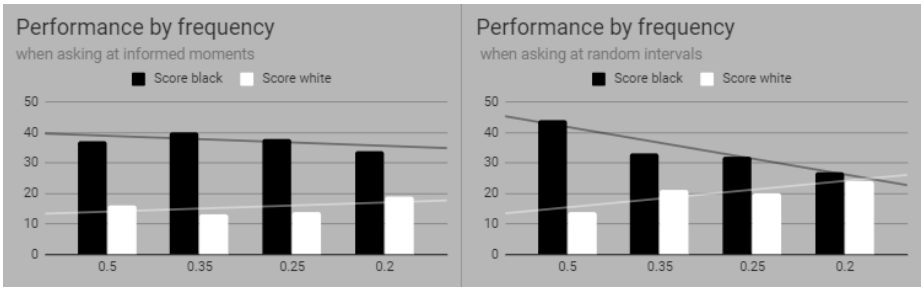

Figure 11: Difference in scores between informed and random condition.

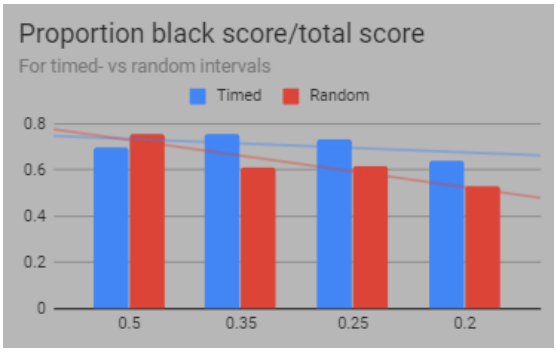

Figure 12: Difference in black's proportion of total points between informed and random condition.

Fitted regression lines show coefficients of 7.143 for the informed condition, and 52.38 for the random condition.

As can be observed from the figure 11, in addition to above regression coefficients the score of black goes down more drastically with the frequency for the random condition as for the informed condition. That makes sense, because with $f = 0.5$, the random moment at which the agent asks for help has more probability of having overlap with a moment at which it would have asked if it were informed, compared to lower values of f.

Additionally, the proportion of the total score black has goes down faster for the random condition (coefficient = 0.6801) as opposed to the timed condition (coefficient = 0.1151), as can also be seen from figure 12.

Using regression analysis, the two slopes can be compared for the timed and random conditions respectively. Visually it is clear that they differ. We wish to show that the slope of score for black, and perhaps proportion of black's score, depends on the condition:

	Timed		Random		Analysis		
	Coefficient	Error	Coefficient	Error	t	Df	p
s_b	7.143	12.37	52.38	9.736	2.87	4	0.0453
p_b	0.1151	0.249	0.6801	0.1673	1.88	4	0.1328

Table 6: Results of regression analysis, testing significance of difference between two regression coefficients.

To calculate t, Df, the following formulas were used [16]:

$$t = \frac{b_1 - b_2}{\sqrt{s_{b_1}^2 + s_{b_2}^2}}. \tag{1}$$

$$Df = n_1 + n_2 - 4. \tag{2}$$

where b_m is the coefficient of the regression model m, s_{b_m} is the standard error of the regression model's coefficient b_m, and n_m the sample size of model m.

- For black's tournament score s_b, the difference between coefficients for the two conditions was **significant**: $p = 0.0453 \leq 0.05$, with a t-test score of 2.87.
- For black's proportion of the total score p_b, the difference between coefficients for the two conditions was **not significant**: $p = 0.1328 > 0.05$, with a t-test score of 1.88.

The score for black is significantly affected by the condition, therefore it is possible to conclude that the time at which the `CooperativeAgent` asks for help has an effect on the performance. This effect was not found significant for the proportion of the total points that black had obtained during the experiments. This would have further supported the claim that informedness or timing of

asking has an effect on performance.

Hypothesis: H_0 can be **rejected**, as the condition {timed, random} has a significant effect on the score and therefore, performance, of the agent.

5 Discussion

5.1 Implications

From the obtained results and respective conclusions that the effects of both help-asking frequency on performance, and moments at which one ask on performance, were significant, it follows that performance can be improved if an artificial agent asks for help from an outside expert.

This implies that in collaborative systems, performance can increase, which extends to forementioned domains such as construction, healthcare and search and rescue. In such domains, dependability, efficiency and safety factors might improve if the artificial agent part of the collaborative system is able to ask for help in states of uncertainty.

Furthermore, since the formalization of uncertainty used in the exploration of the research questions of this thesis had an effect on performance (compared to asking at random intervals), it is implied that there exist formalizations of uncertainty which allow an agent to ask for help at times that make a difference.

It might prove useful to equip artificial agents in collaborative scenarios with mechanisms that allow it to determine when it is better to hand over control. Doing so can facilitate performance of collaborative systems in all manner of different domains.

5.2 Future research directions

From the results of this thesis, several interesting topics of further research come forth:

- Experiment with large sample of human players and compare performance with expert agent scenarios.

  It could be interesting to see what the contribution of human players could be to the performance of the collaborative agent. Additionally, it could be interesting to see whether really good chess players produce a significantly different level of performance compared to less-experienced players, compared over different frequencies. Perhaps their insights provide such good quality moves that can be built upon further in the game.

- Experiment with different qualities of helper agents.

It might also be interesting to look into changing the quality of the helper agent, also in conjunction with above mentioned future research topic. The effects of different qualities of helper agents on the evoked emotions of human collaborators would be interesting as well.

- Run with different, faster, or better performing underlying algorithms

  Additionally, it could be very interesting to test different algorithms, both for the autonomous agent as for the collaborative agent. Some interesting alternatives to Minimax with Alpha-beta pruning, or tree-search algorithms in general, include genetic algorithms [17] and reinforcement learning algorithms [18]. Aside from performance, faster algorithms could also facilitate running much more simulations, aiding in collecting more samples such that variance can be reduced, and more robust conclusions can be drawn from the gathered data.

5.3 Conclusion

In this thesis, a Chess game was implemented, along with two agents that can play against each other. Both are based on Minimax tree-search with alpha-beta pruning, but the collaborative agent can ask for help from either a human, or a better performing agent. That decision is based on frequencies of maximally evaluated game states that flag a state of the game as uncertain.

Experimental results show that the frequency influences the performance of the collaborative agent. Additionally, results show that the timing is of importance as well to the performance, while an agent that asks for help at random intervals performs worse than an agent that does so at moments of uncertainty.

References

[1] Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh Rao, and Minoru Asada. Initiative in robot assistance during collaborative task execution. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, pages 67–74. IEEE Press, 2016.

[2] Johannes Fürnkranz. *Machine Learning and Game Playing*, pages 633–637. Springer US, Boston, MA, 2010.

[3] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.

[4] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and autonomous systems*, 42(3-4):271–281, 2003.

[5] Robin R Murphy, Dawn Riddle, and Eric Rasmussen. Robot-assisted medical reachback: a survey of how medical personnel expect to interact with rescue robots. In *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No. 04TH8759)*, pages 301–306. IEEE, 2004.

[6] Amos Freedy, Ewart DeVisser, Gershon Weltman, and Nicole Coeyman. Measurement of trust in human-robot collaboration. In *2007 International Symposium on Collaborative Technologies and Systems*, pages 106–114. IEEE, 2007.

[7] Robin R Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):138–153, 2004.

[8] K. Kosuge, M. Sato, and N. Kazamura. Mobile robot helper. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 583–588 vol.1, April 2000.

[9] Robert Sparrow. The march of the robot dogs. *Ethics and Information Technology*, 4:305–318, 12 2002.

[10] Hayley Robinson, Bruce MacDonald, Ngaire Kerse, and Elizabeth Broadbent. The psychosocial effects of a companion robot: a randomized controlled trial. *Journal of the American Medical Directors Association*, 14(9):661–667, 2013.

[11] Amir Aly, Sascha Griffiths, and Francesca Stramandinoli. Metrics and benchmarks in human-robot interaction: Recent advances in cognitive robotics. *Cognitive Systems Research*, 43:313–323, 2017.

[12] Peter A Hancock, Deborah R Billings, Kristin E Schaefer, Jessie YC Chen, Ewart J De Visser, and Raja Parasuraman. A meta-analysis of factors affecting trust in human-robot interaction. *Human factors*, 53(5):517–527, 2011.

[13] Deborah R. Billings, Kristin E. Schaefer, Jessie Y.C. Chen, and Peter A. Hancock. Human-robot interaction: Developing trust in robots. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 109–110, New York, NY, USA, 2012. ACM.

[14] World Chess Federation FIDE. Laws of chess: For competitions starting from 1 july 2014 till 30 june 2017. *FIDE Handbook, E. Miscellaneous.*

[15] Grand slam chess final masters bilbao. *retrieved from: https://en.chessbase.com/post/grand-slam-che-final-masters-bilbao.*

[16] Cohen P. West S.G. Cohen, J. and L.S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioural Sciences (3rd edition)*. Lawrence Earlbaum Associates, 2003.

[17] O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu. Genetic algorithms for evolving computer chess programs. *IEEE Transactions on Evolutionary Computation*, 18(5):779–789, Oct 2014.

[18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.