Use of Bayesian Optimisation to maximise player engagement in video games

Candidate number 074936

***Abstract*—Video game development has been on a consistent rise over the decades. Starting from a gameplay with ASCII-coded characters through to heavily-graphical, realistic-looking visuals, video games have evolved to a whole new level. One factor is common in any nature or genre of video games: maximising player engagement. This study adopts Bayesian Optimisation in exploring how player engagement could be maximised by distributing game assets, such as coins and enemies, within a maze. As experiences differ from player to player, there are three player profiles considered in this study: Greedy, Neutral, and Aggressive. In implementing Bayesian Optimisation towards this problem, TPE surrogate modelling and Expected Improvement acquisition function was adopted. The results generated demonstrate that the optimisation retrieves an optimal set of coin and enemy placements, with optimal steps taken by the player, which show error rates of less than 5%. There were other technical improvisations made to fine-tune the code and demonstrate that the implementation goes beyond meeting the project objectives successfully.**

I. Introduction

Video games have evolved over the decades, developed from punch tapes using PDP-1 programming [1] through to graphic-heavy video games with realistic visuals involving extensive hours of hard-coding [2], there is a significant research on how man-hours could be reduced in creating game assets with the help of Artificial Intelligence (AI) and Machine Learning (ML). Tremendous amount of research has gone into automating different elements of game design, including content generation of maps [3], [4], story plots [5], player inventory, and stochastic player-game asset interaction [6]. Whilst the reduction of hard-coding is one advantage, Nakanishi and Zohaib [7] highlight another advantage of tailoring gameplay to player's experience, such that their engagement is maximum. Too easy and the player would be bored, whilst too difficult, the player would find it difficult to complete the game, either case resulting in player not thoroughly enjoying the game.

This study considers this factor of player engagement and the ability to adapt to player's expertise in solving the game by using Bayesian Optimisation. In this study, a square maze design is considered with hard-coded walls as one asset that is fixed, and the game assets, such as coins and enemies change in their position, such that the player takes a certain amount of time to solve the maze - easier the game, lesser time is taken, whereas higher difficulty would increase time spent to reach the end goal. This development would involve three elements - the artificial player profile, the maze design, and Bayesian modelling towards placement of game assets such that the game is neither too easy, nor too difficult.

The structure of this report is as follows:

- Section II describes relevant concepts to this study, including Bayesian Optimisation (BO), Player profiling, maze design, and path-finding within the maze.
- Section III describes the experimental setup and methodology, along with project limitations faced during analysis.
- Section IV provides results of the analyses performed, along with references to literature study performed. This section also reflecting upon, strengths, limitations, and describing how this project successfully met the objectives.
- Section V concludes with project significance and direction to future work.

A. *Project Objective*

This project aims to adopt Bayesian Optimisation to maximise player engagement by distributing game assets appropriately. The success of this project is defined by the implementation unambiguously being able asset placements to demonstrate achieving maximised player engagement as an output.

B. *Project Scope*

- The project scope is restricted to the game design considered as the problem and the limited number of game assets that could be distributed. While the project may describe other game designs explored during problem formation, implementation towards these are considered out-of-scope of this project.
- The implementation is restricted to the problem at hand, with appropriate justifications provided to the approach followed. Implementation by multiple approaches, hyper-parameter tuning, and other surrogate modelling routes, to improve the results is considered out-of-scope, given the tight timeline of this project.
- The approach followed is restricted by domain experience. There may be better approaches beyond BO, however these are not explored, as the focus in this project is to explore the execution of BO towards the video game industry. Any other approach beyond BO are considered out-of-scope.

C. *Project Contribution*

This project explores the use of BO in generating game assets within the design, making it more interactive and tailored to the player experience. The potential contributions of such a project are -

- **Reducing human effort:** This attempt would potentially reduce hard-coding every game asset within game design, and therefore, reducing significant man-hours.
- **Tailoring to audience:** With a factor of player experience and how challenging the player wants the game to be, the focus of the project is to maximise player engagement. In the video game industry, this could be a significant factor in attracting gamers to playing the video game developed with this attribute. Specifically for video games used for educational purposes and learning new skills, players/gamers could consisitently improve on their experience, whilst getting a new gaming experience on every attempt of gameplay.
- **Improving future game versions:** With the era of having game series and a consistent focus on evolving the previous version of video games, a surveying element on gameplay could be incorporated, so that relevant parameters could be fed back for further improvement and continuous learning of diverse player profiles.

II. Background

A. Player profiling

Various literature sources have distinct approaches to player profiles - some have taken surveys on player experience, whereas some have measured player performance during gameplay, and tracking their facial expressions and heartbeat. Whilst it may be argued that the former approach is time-consuming and may yield to inaccuracies in aligning to players in real-time, the latter approach has yielded to promising results with extensive preliminary data available for further analysis. Cowley [8] rightly points this out, however, they also highlight that any real consensus or an empirical approach is challenging to model towards player profiling.

An effective approach defined by Csikszentmihalyi *et al.* [9] is the *Flow* approach, which was further extended by other researchers to list the following elements towards game design and player profiling:

1) A challenging but tractable task to be completed,
2) The task has clear unambiguous goals,
3) One receives immediate feedback on actions,
4) One has complete freedom to concentrate on the task,
5) One feels fully in control,
6) One is fully immersed in the task, no other concerns intrude,
7) One becomes less conscious of the passage of time, and
8) Sense of identity lessens, but is reinforced afterwards.

Csikszentmihalyi [10] also defines the relationship between player profiling and gameplay, and plots this against the challenges the player faces in the gameplay and the skills he possesses to complete tasks successfully. This is plotted as shown in Fig. Nakatsu *et al.* [11] defines a different approach to the *Flow* channel, and forms it as a quadrant, with three other factors into consideration - Apathy (low challenge, low skill), Anxiety (high challenge, low skill), Boredom (low challenge, high skill), whilst the *Flow* quadrant with high challenge and high skill factors.

On the other hand, the latter approach, which is to use player performance and data mining approaches to model player profiles, have been implemented within multiple high-budget, commercial games [12], such as *Max Payne* and *Prey* [13]. An interesting approach of using data mining was by Johansson *et al.* [14], where Poker rules were studied and implemented towards a genetic programming-based, black-box extraction algorithm called G-REX, in order to model player profiles. Another approach using classification algorithms towards artificially generated player agents was attempted by Thawonmas *et al.* [15], wherein player profiles were classified by features defined within the dataset.

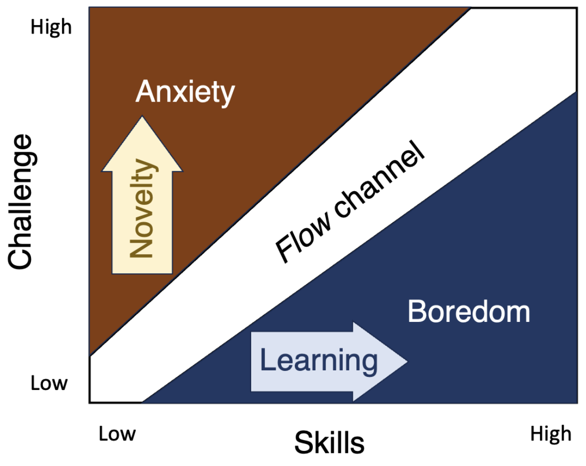


Fig. 1. *Flow* channel: Three player experience dimensions {Anxiety, *Flow*, Boredom} with two modes of perceptual change in the gameplay {Novelty, Learning}[8], [10]

In this project, however, a novel approach to adopting Bayesian Optimisation is attempted to distribute game assets, such that the gameplay follows the *Flow* channel.

B. Game design

In framing the problem, the type of game where BO could be applied to optimise game assets were studied. There were multiple literature sources where either the game assets of different nature were studied, such as selecting weapon inventory, generating game assets by using Procedural Content Generation (PCG) [16], [17], [18], generating alternative storyline, on the basis of dynamic difficulty adjustment (DDA). There has been a considerable amount of work done towards DDA, with approaches ranging from probabilistic models, single and multi-layered perceptrons, dynamic scripting, and reinforcement learning [7]. Additionally, PCG is, in itself, an enormous field of study in generating graphic-intensive assets, such as mountains, terrains, roads, trees, buildings, and floor plans [16]. These assets are, otherwise, hard-coded and are time-consuming, with little to no consideration of tailoring design aspects to player's experience and gameplay approach. As generation of such assets would require a bigger project scope than this study, these were not considered towards framing the problem towards this project. Instead, simpler problems were considered.

One such problem considered was an artificial chess game; This game meets the eight requirements listed by the *Flow* framework. Different player styles were considered towards this problem, namely,

- **Ambitious:** A player would want to defeat the king and end the game as soon as possible, with minimum number of steps.
- **Aggressive:** A player would want to defeat all the opponent entities, and then proceed to defeat the king to end the game.
- **Careful:** A player aims to not have any of its entities defeated ideally, whilst not caring too much about how long it takes to complete the game.

The objective would be to model game assets and player moves such that the time taken falls within the *Flow* channel, for example, the artificial opponent would resist an ambitious player's approach to defeat the king and challenge the player to take multiple approaches to reach its goal, in order to ensure the game is not too easy and too boring. On the other hand, if a careful player only cares about protecting their game entities, then, the game could prolong and take a very long time to complete, thus making the game too difficult. Whilst this was a very interesting idea, one of the major drawbacks towards this game development was its time-consuming aspect, such as -

- Each game entity would have to programmed to appropriately move as per the chess rules.
- Model the artificial opponent, such that it resists the player as per the player's experience level.
- There are multiple factors that would affect the game duration (i.e., number of steps taken) - the movement of six different entities and the situations where the king is obstructed by entities. This would lead to a complex game problem when implementing BO towards number of steps.

Secondly, the player modelling with the data mining approach would have been difficult with no data present on game moves and the situations that may develop in tackling the enemies, which are enormous (10^{111} combinations) [19]. Thus, this game was eventually disregarded in this study.

The second option towards game design was to consider a maze design, where game assets are coins and enemies that are randomly distributed. The maze also consists of walls that would obstruct the player to traverse a minimum possible path to their end goal. The gameplay was considered as follows -

- The player starts from a specific start point, $(0, 0)$, with 0 coins and a health status to full.
- When the player is faced with an enemy, a random generator suggests whether the player successfully defeats or is defeated by the enemy, in order to defeat the enemy. In the event where an enemy defeats the player, the player starts over from the origin with number of steps remaining the same as the last step taken.
- When the player traverses through a coin, his coin score increases to 1.

The objective function should be defined such that the player traverses through the maze such that it is not too easy, and therefore, the player does not take minimal number of steps to reach the end goal, and is not too difficult, and therefore, the player does not take maximum number of steps to reach the end goal. As this game seemed practical and straightforward to solve by an artificial player, this problem was considered as a part of this study.

C. Bayesian Optimisation

Bayesian Optimisation is a black-box function in solving an optimisation problem, which is based on Bayes' theorem. The Bayes' theorem calculates the probability of an event occurring given a specific condition, which is enabled by a set of predicted probabilities by incorporating new information. The Bayes' theorem is mathematically defined as [20]-

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)} \tag{1}$$

where $P(A)$ is the probability of event A occurring, $P(B)$ is the probability of event B occurring, $P(B|A)$ is the probability of event B occurring given event A has occurred, and $P(A|B)$ is the probability of event A occurring given event B has occurred. In context to statistical analysis, Bayes theorem is expanded to finding unknown parameters with prior knowledge, as opposed to what is called a *frequentist* approach. A *frequentist* approach considers probability of an event occurring without any prior conditions or hypotheses. An example of this is the tossing of coin for n times, where previous history of the coin being tossed does not affect the probability of the coin being tossed for the current iteration [21].

1) Why choose a Bayesian approach for this problem over Frequentist?: Towards this problem, the evidence or hypotheses plays a crucial role in defining the posterior on whether the player of a given profile achieves maximum engagement during gameplay, and under what circumstances. Bayesian approach in this respect enables *learning* what kind of asset distribution leads to maximum player engagement. When a frequentist approach is used towards such a problem, every path traversal would be treated as an independent iteration, with no learning of where the maximum player engagement is achieved. A detailed explanation of several Bayesian methods is provided by Williamson *et al.* [22].

2) Steps in implementing BO: Bayesian Optimisation considers three major steps -

1) **Sampling:** The sampling of observations is performed by random distribution methods, such as Latin Hypercube Sampling (LHS).
2) **Surrogate modelling:** A surrogate model is a probabilistic model that is primarily used to tailor towards the sampling structure, reducing the number of evaluations, and thus speeding up the search of potential candidates [23]. A commonly used probabilistic model is Gaussian Process (GP) towards performing BO [24], [25].
3) **Maximising Acquisition Function:** An acquisition function is a utility function that encodes the exploration vs exploitation trade-off [25], which is maximised towards finding the new query point in searching for candidates when performing BO [26]. Exploitation, in

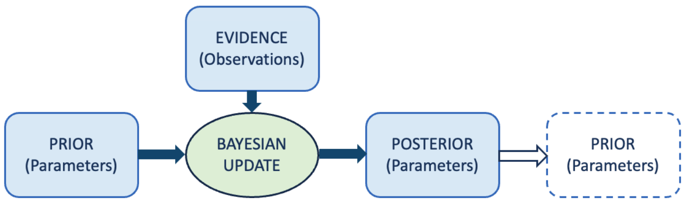
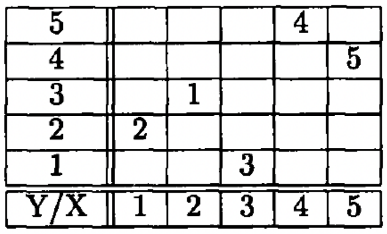


Fig. 2. Flow diagram of a typical Bayesian modelling process



5				4	
4					5
3		1			
2	2				
1			3		
Y/X	1	2	3	4	5

Fig. 3. An example of LHS [30]

the context of BO, is to search near good observations, and exploration is to search unseen regions [27].

D. Sampling

Sampling is the first step to implementing BO. It is crucial to select an appropriate sampling approach for a given problem, as this effectively results in searching for optimal candidates in the least number of simulations. In simpler words, the quality of black-box function evaluations are improved [28].

1) Sampling for clustering problems: For clustering problems, Morishita *et al.* [29] assess the performance of their optimisation by using random sampling, sampling based on D-optimality, Partitioning around mediods (PAM), density-based spatial clustering of applications with noise (DBSCAN) and their combinations.

2) Latin Hypercube Sampling: Another sampling approach is the Latin Hypercube Sampling (LHS). In a problem consisting of a two-dimensional search space, which is the problem type in this study, the LHS approach generates random samples such that each index of the axis has a value assigned [30]. An example is provided by Cheng *et al.* [30] in Fig. 3 - where each of the columns is assigned with random placements.

3) Sobol Sampling: Unlike random and LHS sampling, Sobol sampler considers distributing candidates uniformly across the search space. [31].

4) Thompson Sampling: An interesting sampling method described by Kandasamy *et al.* [32] was synchronous and asynchronous Thompson Sampling (TS). TS is a randomised strategy for sequential decision-making under uncertainty. The basic difference on synchronous ans asynchronous TS is that the former uses information from first three evidences, whilst the latter only considers first evidence when *batches* are set [32].

E. Surrogate modelling

A surrogate model (also known as a Kriging model) when fitted onto an expensive model, facilitates reaching optimal output data by computing marginal probability density function [33]. This is preferred due to its reduced computational cost and easier achievement of optimality [34], [35]. Some of the approaches of fitting a surrogate model are described below -

1) Gradient-enhanced Kriging: Gradient-enhanced Kriging (GEK) is one of the prominent approaches in surrogate modelling towards solving Engineering problems across varying dimensions (2D-20D) [36], [37]. GEK is defined as the extension to kriging with incorporation of gradient information, which effectively improves the accuracy of prediction for a given number of samples [37]. There are two ways to incorporate gradient information, which are classified as direct GEK and indirect GEK methods [38], [39]. The direct method (as the name goes) involves directly including gradient information in the kriging equation system, whilst an additional parameter of finite different step size is to be determined for exploiting gradient information towards indirect GEK methods [40], [41], [42], [35], [43].

2) Radial Basis Function: Apart from use of Gaussian Process (GP), a novel approach of using Radial Basis Function (RBF) as a surrogate model was introduced by Luo *et al.* [44]. They mention that this approach overcomes the challenges of GP when dealing with expensive functions. RBF is known to be widely used for modelling and controlling nonlinear systems due to its universal approximation ability and structural simplicity [45]. Luo *et al.* [44] show RBF as a neural network setup, where the target function is fed into the network during training with predictions of y obtained from the network during testing. The acquisition function used in this study was Expected Improvement (discussed later in section II-F1). They demonstrate the comparison results of single-GP and single-RBF, where the comparison shows RBF outperforming GP surrogate models by almost 5 times.

3) Random Forest: An interesting study by Asritha [46] was to compare Random Forest regressor to be implemented in place of kriging methods towards BO. Random Forest is a supervised ML method, which creates an ensemble model that learns simple decision rules gathered from data features and predict the target value [47]. Dasari [48] describes the process of implementing Random Forest as a surrogate model as follows -

- From a dataset $\mathcal{D}$, draw a bootstrap sample $\mathcal{D}'$, with replacement for each tree construction.
- Build a tree $\mathcal{T}$ using the bootstrap sample, at each node, choose the best split among a randomly selected subset of number of features for tree splits. The tree is constructed until no further splits are possible or reaching given Node size limit.
- Repeat the previous step until user-defined number of trees (limit) is reached.

This was implemented towards an aerospace use case in designing a turbine rear structure [48].

4) Tree-structured Parzen Estimator: Several literature sources [24], [23] describe Tree-structured Parzen Estimator (TPE) to be effective in problems over GP Regression in many instances. As derived by the name, TPE has tree-structured search space and use Parzen estimators (also known as Kernel

Density Estimators). TPE makes the following assumptions [27] -

$$p(\mathbf{x}|y,\mathcal{D}) = \begin{cases} p(\mathbf{x}|\mathcal{D}^{(l)}), (y \geq y^{\gamma}) \\ p(\mathbf{x}|\mathcal{D}^{(g)}), (y \leq y^{\gamma}) \end{cases} \tag{2}$$

where $\mathcal{D}$ is the dataset consisting of samples, y^{γ} is the top-γ-quantile objective value, $\mathcal{D}^{(l)}$ is the better group of samples, whereas $\mathcal{D}^{(g)}$ is the worse group of samples. Tiao *et al.* [24] mentions that TPE has limitations, however, these were not explicitly mentioned in their paper. Nevertheless, this algorithm is considered for comparison towards our implementation.

F. Acquisition Function

Acquisition functions are *enrichment (or infill) strategies*, which post-process the surrogate model. The aim in most cases is to maximise the acquisition function in order to recognise global optimality [49], [45]. A few examples are described below -

1) Expected Improvement: One of the most popular acquisition functions used in implementation of BO is the Expected Improvement (EI) [50]. The expected improvement is computed as follows [51] -

- Let $f_{min} = min(y^{(1)}...y^{(n)})$ be the best possible objective value. However, this is unknown at this stage.
- This is presented by random variable Y with a certain mean and standard deviation.
- A density function within samples of Y are calculated to determine the probability of whether f_{min} is better than y_i (sample of Y).
- The expected value to obtain EI is calculated as follows -

$$E(\boldsymbol{I}(x)) = E(max(f_{min} - y_i), 0) \tag{3}$$

2) Upper Confidence Bound: Upper Confidence Bound (UCB) is a preferred acquisition function towards many Engineering applications [52], [53]. With UCB, the exploitation vs. exploration tradeoff is straightforward and easy to tune via the parameter λ. UCB is a weighted sum of the expected performance captured by exploitation, $\mu(x)$, of the GP, and of the uncertainty (exploration), $\sigma(x)$, captured by the standard deviation of GP. UCB is mathematically defined as [54]-

$$\alpha(x;\lambda) = \mu(x) + \lambda\sigma(x) \tag{4}$$

G. Hyperparameter Tuning

Hyperparameters are the variables of the algorithm that control its whole behaviour. It affects its speed, resolution, structure, and eventually, performance. With algorithms having multiple hyperparameters, it could be challenging to try all combinations to find the best set - therefore, hyperparameter tuning is of crucial importance [55]. Beyond manual choosing of hyperparameters, hyperparamter optimisation is performed prior to BO in order to yield best set of solutions. This is usually represented in the equation form as -

$$x^* = argmin_{x \in X} f(x) \tag{5}$$

where, x^* is the set of hyperparameters that yields the lowest value of the score, $f(x)$ represents an objective score to minimize— such as Root Mean Square Error (RMSE) or error rate— evaluated on the validation set, and x is present within domain X [56].

III. Methodology

A. Maze design and gameplay

The maze design is a square matrix of size n. The start position of the player is $(0,0)$ and is intended to traverse to the end position is $(n-1, n-1)$. A path-finding algorithm, such as A-star, is adopted for the artificial player to reach the end goal. The square maze consists of static walls, which are randomly distributed with `wall_density`, which is the ratio of wall-to-maze size. For example, if `wall_density` is 0.3, the maze would consist 30% of wall co-ordinates. The sampling approaches considered in distributing game assets are random and LHS. There were multiple variations tried to see how the objective function change with associated parameters, such as maze size, avoiding nodes, preferred nodes, and the gameplay itself. One variation of gameplay (as described in section II-B) involved adding an extra element of *what if the player is unable to defeat the enemy*. A situation was set up by utilising a random number generator from 0 to 10, if the player scores a random number of 5 or less, the player loses against the enemy, and has to start the game over. In such a scenario, the number of steps is dependent on the randomness and the likelihood of the player defeating the enemies e number of times. The likelihood of the player successfully defeating all enemies in one go without starting over is $(0.5)^e$, which means that if there are 50 enemies in a 100×100 maze to defeat, the probability of successfully defeating all enemies in one go is $(0.5)^{50} = 8.88\times10^{-16}$, which is essentially *noiseless* number of steps by the artificial player. This approach is particularly inaccurate, as in real-world, a player might have a constant gameplay experience, and therefore, the ability to defeat the enemy would be more deterministic than random, unless the difficulty to defeat enemies differs at each maze. Therefore, this variation was removed and the player was modelled such that they are well-versed to defeat the enemy by default.

B. Player profiling

In section II-A, different approaches to player profiling were explored. Most of these approaches required human input or a dataset created from human-involved data. This project takes a simpler approach of describing profiles and their affinity towards a specific game asset. There are three player profiles in consideration -

- **Greedy:** A greedy player is interested in collecting as many coins as possible and is not interested in defeating enemies, before it ultimately reaches the end goal. Where the greedy player has no other option than to collect the coin before defeating the enemy, as an exception, the enemies are defeated.
- **Neutral:** A neutral player is interested in both collecting coins and defeating enemies before it reaches the end goal. Another approach to modelling a neutral player would be to neither collect coins nor defeat enemies, but to make way to the end goal, however, this would result

in the following limitations towards the project objective -

- The placement of game assets will not have any significant effect on the number of steps taken, as the player would focus only on reaching the end goal as soon as possible.
- The number of steps would remain constant, given the walls are hard-coded.
- The overall game would be less engaging, as the number of steps would be very less compared to other player profiles.

Therefore, the former approach is chosen.

- **Aggressive:** An aggressive player is interested in defeating as many enemies as possible, ignoring collecting coins, before it ultimately reaches the end goal. Where the aggressive player has no other option than to traverse the maze node to defeat the enemy, the coin in that node is collected automatically.

It is apparent that the neutral player would require more time when it has to collect coins *and* defeat enemies. Therefore, it is reasonable to assume that the number of steps for neutral player would be much higher to the greedy or the aggressive player.

C. Bayesian modelling

Considering the project objective, the aim of this implementation is to maximise player engagement. This is achieved by estimating the number of steps within the *Flow* channel in Fig. 1 for all the three player profiles. In this project, as the player experience is constant, the *Flow* channel would imply number of steps being not too few so as to be within the *boredom* range, whilst not too many so as to be within the *anxiety* range. The objective function considers the mean of the number of steps and aims to minimise the difference between mean number of steps and estimated number of steps, which is defined in equation 7.
The parmeters in consideration are coins and enemies. The objective function was initially defined as -

$$\bar{x} = \frac{[2 \times (\bar{n}_{steps})] + [10 \times (\bar{n}_{coins}) + (\bar{n}_{enemies})]}{3} \tag{6}$$

where, $\bar{n}_{steps}$ is the mean of number of steps taken, $\bar{n}_{coins}$ is the mean of number of coins collected, and $\bar{n}_{enemies}$ is the mean of number of enemies defeated by the player. The magnitudes 2 and 10 are applied as weights with respect to the significance of certain model parameters in defining the objective, i.e., player engagement. The value, 3, was defined as per the number of parameters. These also account for player engagement being modified for different player profiles. $\bar{x}$ is the factor in the minimisation problem, which is essentially the mean of 1000 iterations for the above parameters.
However, these did not have any significant effect on the results. Therefore, the parameter in consideration was reduced to the number of steps, s.

$$obj.func. \in [0.90\bar{s}, 1.10\bar{s}] \tag{7}$$

However, this was improvised upon due to achieving promising results. This is further described in section IV-D. A graphical representation of mean steps calculated and drawn as a normal distribution is presented in Fig. 4.

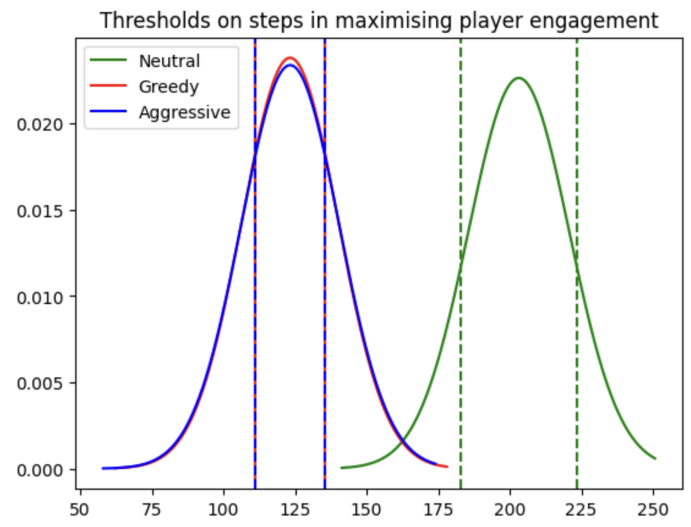


Fig. 4. Thresholds are defined as $\pm 10\%$ of mean steps calculated by 1000 iterations. This is a representative example, as this was further improvised, as described in section IV-D.

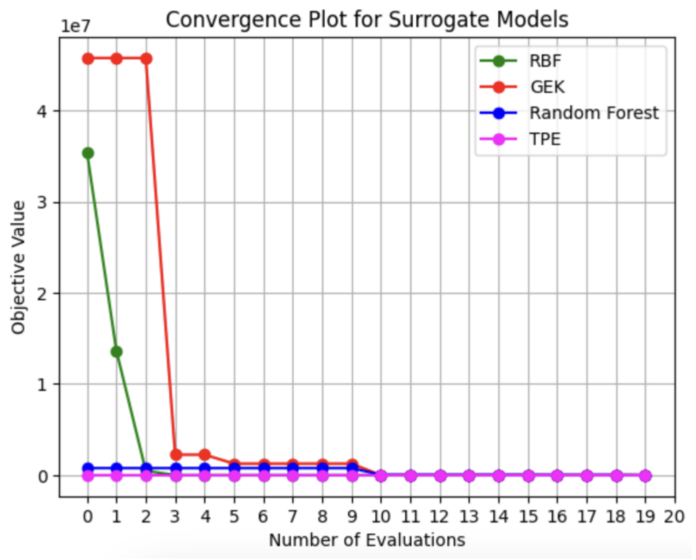


Fig. 5. Plot showing surrogate models reaching convergence $y = 3x^4 - 5x^3 + 7x^2$. TPE reaches convergence right from the first iteration.

1) Sampling: The types of sampling used in this study are `random` and `LHS`. These samplers are applied to the game assets (coins and enemies). The wall density is the ratio of number of nodes assigned as walls to all number of nodes within the maze. The wall density is randomly distributed. The Sobol sampler is not considered due to its uniformity attribute, as this might not necessarily be relevant when distributing game assets. Thompson samplers are also not considered, as these were applied to parallel BO [32], whilst in this study, a more simplistic BO setting is adopted.

2) Surrogate modelling: Selecting a surrogate model approach is a crucial aspect towards implementation of BO. In assessing all the surrogate models discussed in section II-E, a

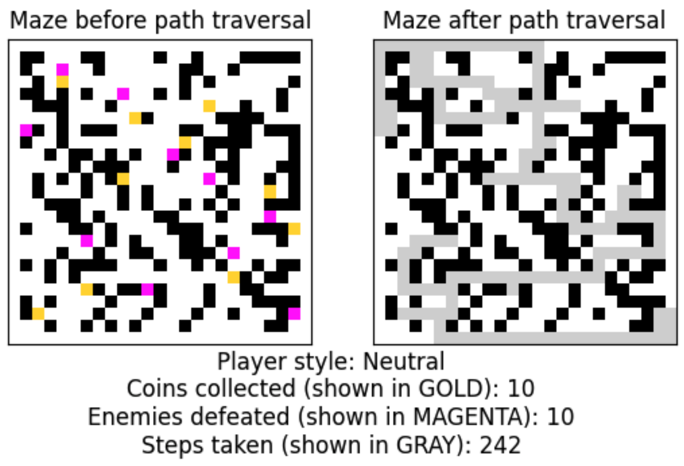

Maze before path traversal
Maze after path traversal
Player style: Neutral
Coins collected (shown in GOLD): 10
Enemies defeated (shown in MAGENTA): 10
Steps taken (shown in GRAY): 242

Fig. 6. A maze of size (n) = 25, with number of coins = 10, number of enemies = 10, wall density = 0.4, is generated. For a Neutral player, the A-star algorithm focusses on both collecting coins and defeating enemies, and eventually reaching the end goal $(n-1, n-1)$

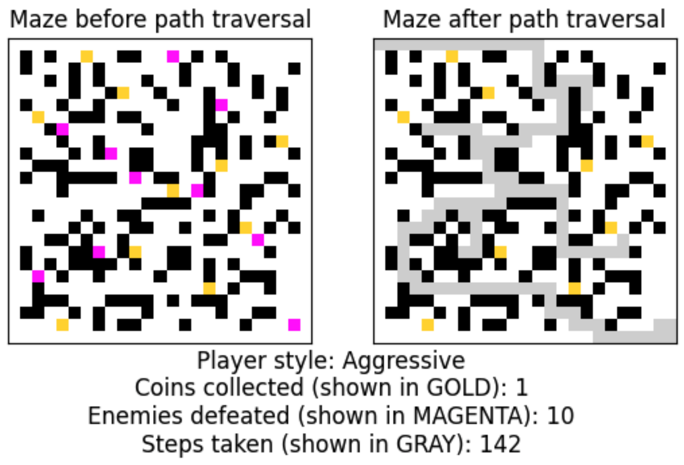

Maze before path traversal
Maze after path traversal
Player style: Aggressive
Coins collected (shown in GOLD): 1
Enemies defeated (shown in MAGENTA): 10
Steps taken (shown in GRAY): 142

Fig. 7. A maze of size (n) = 25, with number of coins = 10, number of enemies = 10, wall density = 0.4, is generated. For an Aggressive player, the A-star algorithm prioritises defeating enemies over collecting coins, and eventually reaching the end goal $(n-1, n-1)$

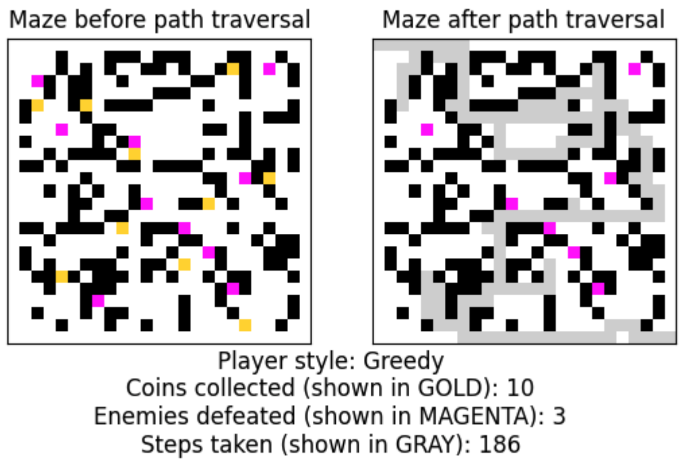

Maze before path traversal
Maze after path traversal
Player style: Greedy
Coins collected (shown in GOLD): 10
Enemies defeated (shown in MAGENTA): 3
Steps taken (shown in GRAY): 186

Fig. 8. A maze of size (n) = 25, with number of coins = 10, number of enemies = 10, wall density = 0.4, is generated. For a Greedy player, the A-star algorithm prioritises collecting coins over defeating enemies, and eventually reaching the end goal $(n-1, n-1)$

toy problem $y = 3x^4 - 5x^3 + 7x^2$ was used to compare these models in achieving convergence. These are plotted in Fig. 5. It is evident that TPE performs the best in reaching optimality as compared to other models. Therefore, this surrogate model was used towards implementation of BO in this study.

3) Acquisition function: The acquisition function considered in this study is Expected Improvement, as it was easy to implement towards this type of a problem.

D. Experimental setup

The project code is written in Python. The code is broken down into the following parts:

1) **Maze design:** The maze class consists of the following variables:
   - Maze size - Integer value, defines n for the maze of size n, such that a NumPy 2D array is returned with value 0.
   - Wall density - Floating point value ranging from 0 to 1, this is calculated as the ratio of wall nodes to all the present nodes within the maze. For example, if the wall density is 0.3, 30% of the maze is filled with walls. These are randomly distributed within the maze.
   - Number of coins - Integer value; Takes c number of coins and randomly allocates them locations within the maze.
   - Number of enemies - Integer value; Takes e number of enemies and randomly allocates them locations within the maze. In most experiments performed, $c = e$, however, the class offers the ability to define them different number of assets, if required.
   - Sampling - Either `'Random'` or `'LHS'` explicitly describes what type of sampling is adopted to randomly allocate the game assets to maze locations.
2) **Path-finding algorithm:** As mentioned, A-star path-finding algorithm is adopted. This is further tailored to each of the player profile. Initially, the 'Greedy' player was inclined to collecting coins over defeating enemies, therefore, they would not explicitly try to avoid enemies, but rather prefer it; Likewise for the 'Aggressive' player style, they would prefer avoiding coins. This was coded as the cost function being slightly higher than a node where their preferred asset is. For example, if the cost function for a *no-asset* node is 1, the cost function of a *preferred* node is 0 (coin for greedy, enemy for aggressive), then the *preferred avoided* node had a cost function of 5 (enemy for greedy, coin for aggressive). This, however, did not work, as the heuristic function could not be anticipated to what values these would generate, as these are a function of the maze design and steps taken as a result of the maze design. Therefore, a simplistic approach was chosen, where an *avoided* node is assigned a cost function of 100 (in case of a wall being present should be avoided), and a *preferred* node was not defined explicitly. This path-finding algorithm worked well, and is further used to solve the maze

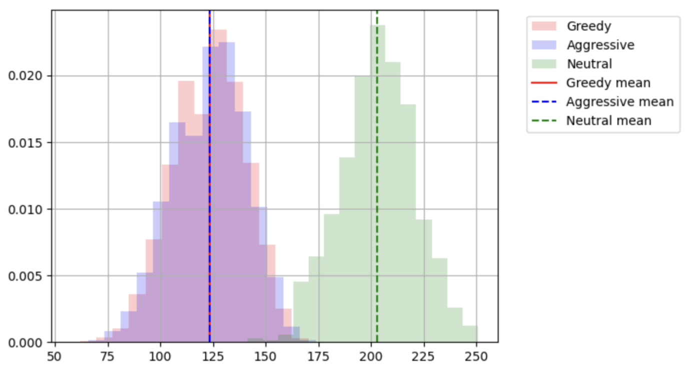


Fig. 9. When running 1000 iterations on steps taken by Greedy, Aggressive, and Neutral player types, the distribution is similar to a Normal distribution, with mean presented by *dotted* lines

problem to retrieve number of steps for corresponding player profiles.

3) **Player profiling:** There are three player profiles set up - Greedy, Neutral, and Aggressive, as discussed in section III-B. These are passed as an argument in the `Player` method. This method calculates coins collected, enemies defeated, and steps taken for a given player profile, and returns these back for further analysis.
4) **Bayesian Optimisation:** BO is implemented from scratch in this study, with explicit definitions of the surrogate model and acquisition function in consideration, with first principles applied, as described in section II.

E. Project Management

1) Code implementation: All versions of code were managed on *GitHub* [57], and were run primarily on *Google Colab*.

2) Task management: Daily and weekly to-do's were assigned on an app called *TickTick*, and were aimed to be completed on the deadline set. If for any reasons these were not met, they were mitigated.

IV. Results

A. Maze design and path traversal

The initial maze generated contains game assets, including walls, distributed either randomly or by adopting LHS. The path-finding algorithm, A-star, is adopted to traverse through the maze for different player styles (Greedy, Neutral, and Aggressive). Examples for different player profiles are shown in Fig. 7, 8, 6.

B. Player Style

The number of steps differ with the type of player and priorities on traversing through game assets. When running n simulations, the number of steps by each player style retrieved exhibits a normal distribution, which is assessed and a mean and standard deviation is retrieved. This is, then, used towards Bayesian modelling in distributing coins and enemies across the maze. An example of the distribution is shown in Fig. 9.

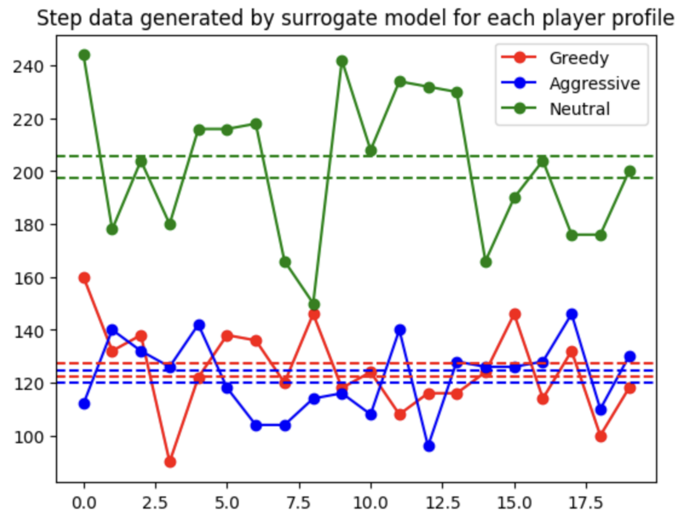


Fig. 10. A graphical representation of surrogate modelling and ±5% maximum player engagement threshold

Player profile	Optimal steps	% diff.
Greedy	124	3.01%
Neutral	206	3.41%
Aggressive	126	4.62%

TABLE I
Optimal values for each player profile and percentage difference from mean step to validate optimality

C. Bayesian Optimisation

As discussed in section III-C, the implementation of BO is divided into three major steps. The approaches to implementing these steps were also discussed. These are explicitly defined in the code. The results obtained are the optimal placements of coins and enemies, and the optimal steps taken by the player of each profile. The asset distribution is shown in Fig. 11, where it could be observed that assets are neither distributed as closer clusters, nor are they too far off to increase the number of steps. Instead, they are relatively balanced across the maze space. The step data retrieved for each player profile by the surrogate model is plotted in Fig. 10. The acquisition function, EI, plays a crucial role in finding the optimised parameters, wherein, these were recognised by maximising the acquisition function. The optimal steps taken by each player profile are tabulated in Table I.
To validate these results to thresholds set in Fig. 4, the percentage difference is also included in Table I. The results show not just meeting the thresholds between ±10%, but was further improvised to optimise steps within ±5% of mean step data.

D. Project improvisations

- The modules to implement BO, such as *skopt* and *hyperopt* were attempted, however, these led to either unsatisfactory results or algorithms and acquisition functions not being explicitly defined. Taking this shortcoming as an opportunity, first principles were studied in further detail and applied from scratch, thus, demonstrating understanding of the concept at its first principles.

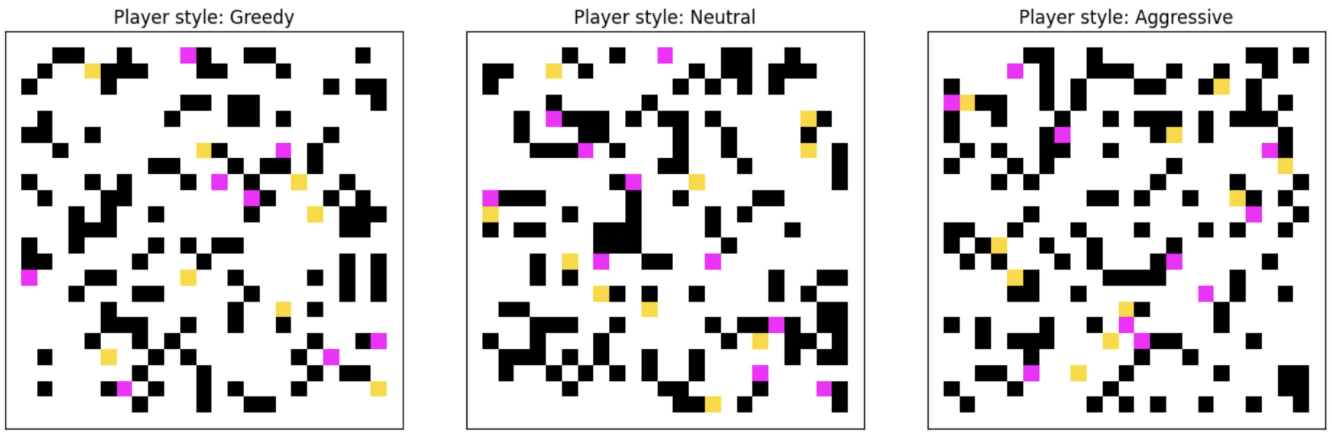


Fig. 11. The maze structure and asset distribution after BO implementation and when optimal asset positions are known

- The project reference provided as a part of this study by the department involved an approach of dividing the maze into quadrants and finding probabilities of coins and enemies being placed within these quadrants. However, this would not have led to an unambiguous demonstration of optimal positions and optimal steps taken by each player profile, which is the prime objective of this project. Additionally, there was no way to validate the probabilities retrieved and to demonstrate how accurate the BO implementation has been. Therefore, an alternative approach of implementing BO from scratch was adopted. This led to meeting the project objective successfully.
- The optimal steps was aimed at $\pm 10\%$ of the mean steps taken by each player profile. However, due to implementation providing promising results, this threshold was slimmed down to $\pm 5\%$. The optimal steps and asset positions were generated surprisingly well within this slender margin.

E. Limitations and challenges

- When implementing BO, the computational cost for 1000 iterations was much higher, especially when these were executed in multiple loops, namely in determining mean steps, generating samples by surrogate modelling, and finding next sample with the aid of EI. Therefore, these were reduced down from 1000 iterations to 100 iterations for sampling and 20 evaluations towards BO implementation. It is anticipated that a more optimal distribution might take place if number of evaluations were increased, as this would lead to learning of more hypotheses.
- The maze design had a tiny flaw, wherein walls were generated randomly. For cases of wall density being higher, there were likely chances that assets were trapped around walls. The code returned a `NoneType` path, which was not helpful. This was rectified by using a lower wall density. Despite of this being an ad-hoc solution, this worked well in implementing BO and retrieving results. On rare occasions, the `NoneType` error is risen. In such cases, the simulations were run again.

V. Conclusion

This study involved making informed decisions at multiple stages of experimental design, from search space (maze) design, to player profiling, and hyperparameters in implementing BO. These were described and justified in section III-D. The end goal (project objective) was achieved by unambiguously retrieving optimal positions of coins and enemies within the maze, along with optimal steps taken by each player profile.
As this is an open-ended problem, i.e., to maximise player engagement in video games, there are several directions for future work towards this study. For instance, the implementation could be extended to various genres of games - an example was the chess game described in section II-B. Other examples include maze pattern with sophisticated game assets such as ghosts appearing out of nowhere and treasures to be collected [58], or role-playing games [59].
In section II-B, other elements to game design where BO could be implemented were discussed - PCG being the most interesting of it. It could be an interesting study to explore graphically-intensive game assets, such as mountains, terrains, road networks, and/or buildings by the BO implementation.
In this study, the player profiles were limited to three. However, this may not be aligned to player modelling that represents the real-world. There are multiple dimensions to player profiling, such as variable player experience when dealing with player scenarios, player mood, emotions, attributes, facial expressions, and gameplay with multiplayer interaction - all of these could form useful features of player profiling. The dataset could be generated by surveying, tracking gameplay through real players, and using computer vision, for instance, to detect player mood and emotions [58].
Imagining the game development industry where hard-coding is reduced to automate generation of game assets could lead to significant time being minimised, and therefore, potentially developing more games than the current process. Additionally, when the video game is tailored to the individual player, the game enhances on gameplay quality. Given these aspects, this domain has a lot of potential for further research and promising contributions to the video game industry.

Declaration

Declaration of Originality: I am aware of and understand the University of Exeter's policy on plagiarism and I certify that this assignment is my own work, except where indicated by referencing, and that I have followed good academic practices.

Declaration of Ethical Concerns: This work does not raise any ethical issues. No human or animal subjects are involved, neither has personal data of human subjects been processed. Also no security or safety critical activities have been carried out.

References

[1] Creative computing. https://worldcat.org/title/1355381274, 1981.

[2] Amanda C. Cote and Brandon C. Harris. The cruel optimism of "good crunch": How game industry discourses perpetuate unsustainable labor practices. *New Media & Society*, 25(3):609–627, 2023.

[3] Julian Togelius, Mike Preuss, and Georgios N Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 workshop on procedural content generation in games*, pages 1–8, 2010.

[4] William L Raffe, Fabio Zambetta, Xiaodong Li, and Kenneth O Stanley. Integrated approach to personalized procedural map generation using evolutionary algorithms. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(2):139–155, 2014.

[5] Jurie Horneman. Procedural level and story generation using tag-based content selection. In *Game AI Pro 3: Collected Wisdom of Game AI Professionals*, pages 451–459. CRC Press, 2017.

[6] Hendrik Skubch. Ambient interactions: Improving believability by leveraging rule-based ai. In *Game AI Pro 360: Guide to Character Behavior*, pages 113–124. CRC Press, 2019.

[7] Hideyuki Nakanishi and Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018:5681652, 2018.

[8] Benjamin Cowley. *Player Profiling and Modelling in Computer and Video Games*. PhD thesis, 01 2009.

[9] Mihaly Csikszentmihalyi and Isabella Selega Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge University Press, 1988.

[10] Mihaly Csikszentmihalyi. *Beyond boredom and anxiety*. The Jossey-Bass behavioral science series. Jossey-Bass Publishers San Francisco, San Francisco, 1st ed edition, 1975.

[11] Ryohei Nakatsu, Matthias Rauterberg, and Peter Vorderer. A new framework for entertainment computing: From passive to active experience. pages 1–12, 09 2005.

[12] Robin Hunicke and Vernell Chapman. Ai for dynamic difficulty adjustment in games. *Challenges in game artificial intelligence AAAI workshop*, 2, 01 2004.

[13] Ben Reeves. Arkane knowledge: Five reasons dishonored fans will love prey, 2016.

[14] Ulf Johansson, Cecilia Sönströd, and Lars Niklasson. Explaining winning poker–a data mining approach. pages 129–134, 12 2006.

[15] Ruck Thawonmas, Ji-Young Ho, and Yoshitaka Matsumoto. Identification of player types in massively multiplayer online games. 2003.

[16] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22, 2013.

[17] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1):19–37, 2021.

[18] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

[19] L.V. Allis. *Searching for solutions in games and artificial intelligence*. PhD thesis, Maastricht University, January 1994.

[20] Adam Hayes. https://www.investopedia.com/terms/b/bayes-theorem.asp, 2023.

[21] Cassie Kozyrkov. Statistics: Are you bayesian or frequentist? https://towardsdatascience.com/statistics-are-you-bayesian-or-frequentist-4943f953f21b, 2021.

[22] J. Williamson, A. Oulasvirta, P. Kristensson, and N. Banovic. An introduction to bayesian methods for interaction design. page 3–80, 2022.

[23] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664. PMLR, 06–11 Aug 2017.

[24] Louis C Tiao, Aaron Klein, Matthias W Seeger, Edwin V. Bonilla, Cedric Archambeau, and Fabio Ramos. BORE: Bayesian optimization by density-ratio estimation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10289–10300. PMLR, 18–24 Jul 2021.

[25] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization, 2020.

[26] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016. Publisher Copyright: © 1963-2012 IEEE.

[27] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance, 2023.

[28] Setareh Ariafar, Zelda Mariet, Ehsan Elhamifar, Dana Brooks, Jennifer Dy, and Jasper Snoek. Weighting is worth the wait: Bayesian optimization with importance sampling, 2020.

[29] Toshiharu Morishita and Hiromasa Kaneko. Initial sample selection in bayesian optimization for combinatorial optimization of chemical compounds. *ACS Omega*, 8(2):2001–2009, 01 2023.

[30] Jian Cheng and Marek Druzdzel. Latin hypercube sampling in bayesian networks. pages 287–292, 01 2000.

[31] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based Rendering: From Theory to Implementation*. GitHub Pages, 2004.

[32] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised bayesian optimisation via thompson sampling. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 133–142. PMLR, 09–11 Apr 2018.

[33] Sascha Ranftl and Wolfgang von der Linden. Bayesian surrogate analysis and uncertainty propagation. *Physical Sciences Forum*, 3(1), 2021.

[34] Raphael T Haftka, Diane Villanueva, and Anirban Chaudhuri. Parallel surrogate-assisted global optimization with expensive functions–a survey. *Structural and Multidisciplinary Optimization*, 54:3–13, 2016.

[35] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[36] Selvakumar Ulaganathan, Ivo Couckuyt, Tom Dhaene, Joris Degroote, and Eric Laermans. Performance study of gradient-enhanced kriging. *Engineering with Computers*, 32(1):15–34, 2016.

[37] Zhong-Hua Han, Stefan Görtz, and Ralf Zimmermann. Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function. *Aerospace Science and Technology*, 25(1):177–189, 2013.

[38] Hyoung Seog Chung and Juan Alonso. Design of a low-boom supersonic business jet using cokriging approximation models. In *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, page 5598, 2002.

[39] Seongim Choi, Juan J Alonso, and Hyoung S Chung. Design of a low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method. *AIAA paper*, 1758:2004, 2004.

[40] Weiyu Liu and Stephen Batill. Gradient-enhanced response surface approximations using kriging models. In *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, page 5456, 2002.

[41] Weiyu Liu. *Development of gradient-enhanced kriging approximations for multidisciplinary design optimization*. University of Notre Dame, 2003.

[42] Julien Laurenceau and P Sagaut. Building efficient response surfaces of aerodynamic functions with kriging and cokriging. *AIAA journal*, 46(2):498–507, 2008.

[43] Richard Dwight and Zhong-Hua Han. Efficient uncertainty quantification using gradient-enhanced kriging. *Proceedings of 11th AIAA Conference on Non-Deterministic Approaches*, 05 2009.

[44] Jianping Luo, Wei Xu, and Jiao Chen. A novel radial basis function (rbf) network for bayesian optimization. In *2021 IEEE 7th International*

Conference on Cloud Computing and Intelligent Systems (CCIS), pages 250–254, 2021.

[45] Yunwen Lei, Lixin Ding, and Wensheng Zhang. Generalization performance of radial basis function networks. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3):551–564, 2015.

[46] Kotha Sri Lakshmi Kamakshi Asritha. Comparing random forest and kriging methods for surrogate modeling, 2020.

[47] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *Forest*, 23, 11 2001.

[48] Siva Krishna Dasari, Abbas Cheddad, and Petter Andersson. Random forest surrogate models to support design space exploration in aerospace use-case. In John MacIntyre, Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 532–544, Cham, 2019. Springer International Publishing.

[49] Luc Laurent, Rodolphe Le Riche, Bruno Soulier, and Pierre-Alain Boucard. An overview of gradient-enhanced metamodels with applications. *Archives of Computational Methods in Engineering*, 26(1):61–106, 2019.

[50] Matthias Schonlau, William Welch, and Donald Jones. *Global versus local search in constrained optimization of computer models*, volume 34, pages 11–25. 01 1998.

[51] Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998.

[52] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[53] Sandeep Pandey and Christopher Olston. Handling advertisements of unknown quality in search advertising. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[54] Stathis Kamperis. https://ekamperi.github.io/machine%20learning/2021/06/11/acquisition-functions.html#upper-confidence-bound-ucb, 2023.

[55] Maria Gorodetski. Hyperparameter tuning methods - grid, random or bayesian search? https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399, 2021.

[56] Will Koehrsen. A conceptual explanation of bayesian hyperparameter optimization for machine learning. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f, 2018.

[57] Dushyant Khatri. https://github.com/vedkhatri/masters_project, 2023.

[58] Krzysztof Kutt, Dominika Drażyk, Laura Żuchowska, Maciej Szelażek, Szymon Bobek, and Grzegorz J. Nalepa. Biraffe2, a multimodal dataset for emotion-based personalization in rich affective game environments. *Scientific Data*, 9(1):274, 2022.

[59] Henrik Schoenau-Fog et al. The player engagement process- an exploration of continuation desire in digital games. In *Digra conference*, 2011.