Bad Bohemians: Generating Whodunnits by Genetic Algorithm

January 12, 2023

Contents

1 Introduction

In a retrospective on *The Digital Antiquarian Blog* (2013), Jimmy Maher posits that mysteries have had such an enduring appeal to game designers because no other genre "so obviously sees itself as a form of game between reader and writer". If the mysteries of traditional, static media are to viewed as games, then (at the risk of being overly literal) what are their mechanics? Agatha Christie's *Murder on the Orient Express* is rightly beloved as a detective novel, but surely lacks two key features as a *game*:

1. *Deduction* To write a whodunnit is to create a singular win condition for the player: the who of a case becomes all that matters, rather that the more intricate (and perhaps more unfair) questions of how and why. However, with a cast of n suspects, the passive player who makes no deductions and guesses the culprit at random will still win with probability $1/n$ (or at least $1/2^n$, allowing for multiple murderers...)

2. *Verification* The player will only ever get one attempt at solving a given mystery; to read to the end and verify if their best guess is correct is to spoil a whodunnit forever. While there is a lot of satisfaction to be found in re-reading a detective novel and seeing the clues hidden in plain sight, can it ever provoke the same deductive satisfaction as the first time around?

Understanding these as inherent limitations of static literature, the appeal of the mystery game may be its existence as an open problem for the ludic medium, as a holy grail for procedural storytelling: **a retriable, replayable case, solvable only with the active participation of the player.**

'Einstein's Puzzle' (also known as 'The Zebra Puzzle'; the link to Einstein may be pure conjecture) is a well-known form of brainteaser. The player is given a list of 5 nationalities, 5 houses, 5 beverages and cigarette brands of choice, and 5 pets, with just enough clues that they can use logic to match exactly one element from each category to exactly one of 5 characters. While not strictly *about* being a detective, these so-called Einstein puzzles stand out as a meaningful test of one's deductive capabilities in a medium that all-too-often fails to make the player think for themself (Game Maker's Toolkit 2017).

This report will discuss *Bad Bohemians*, a text-based application that uses the same design principles. Since the narrative potential of the logic puzzle has already been realised to great success - with *Return of the Obra Dinn* (Pope 2018) and *The Case of the Golden Idol* (Color Gray Games 2022) both released to much critical acclaim - *Bad Bohemians* aims to introduce a more procedural element. Using a genetic algorithm, the application creates Einstein puzzles from a set of pre-written narrative beats, allowing the player to work towards a new solution each time they play; it is an AI designed to solve the creative problem given in bold above.

Popularised by the work of John Holland (1992), genetic algorithms can be a rewarding tool in the creative arts, seeing use in the generation of everything from 'evolutionary music' (Gibson & Byrne 1991) to 3D models (Haubenwallner, Seidel & Steinberger 2017). With regards to procedural storytelling, one particularly interesting essay (Compton 2019) explains that, "genetic algorithms do not generate content, as you still need a generator, but they *guide* that generator toward more constraint-fulfilling and desirable-property-producing content". This aligns perfectly with *Bad Bohemians*' own requirements: the challenge lies not in generating sentences, but in mixing and matching them until arriving at a mystery with one unique solution.

The application also relies heavily on its inference engine, an underlying rules-based system for identifying which deductions a player can or can't make. Expert systems are notoriously inefficient, both in regard to the time they take to code and the speed at which they run. For this project's purposes, however, they are a necessary evil. Even one mistake by the engine can render a puzzle completely unsolvable, so if the rules for when '*a* implies *b*' are hard-coded by a human, then all inferences will come with a first-hand quality assurance.

Bad Bohemians is written entirely in C#, for ease of integration into a future game. This report will focus primarily on its genetic algorithm, being the only learning AI involved. While the inference engine will be discussed when necessary, in the interests of brevity no formal breakdown of its ruleset will be provided; while it may have taken a lot of time to code, one now has the luxury of treating `Logic.cs` as a black box.

2 Methodology

At the highest possible level, *Bad Bohemians* generates its mysteries by first establishing a set of characters and a corresponding set of clues, then running a genetic algorithm to check which combination of clues available will result in the most satisfying puzzle. Ideally, these two stages exist completely independent of one another, so that narrative designers have freedom to experiment with how they frame their mysteries without ever needing to tamper with the AI.[1]

[1] This is largely, but not completely, true of the current implementation: for example, the fact that exactly 1 clue appears per scandal would be a narrative decision that is, at time of writing, still hard-coded into the genetic algorithm.

2.1 Premise and Pre-Processing

Bad Bohemians, for instance, a game about the affairs and internecine squabbles of high society, is framed through the lens of a Victorian gossip column. The game recounts the recent scandals of 5 stock characters (e.g. a highway robber), then challenges the player to match each scandal with a forename, surname, title and county (e.g. Lazarus Sallow, Duke of Hanley Heath); whodunnits, at least in a loose sense of the word.

As such, the application starts by creating a cast of 5 characters out of elements in these five 'primary categories', as well as the 'secondary category' that is gender. Not quite as simple as randomly combining strings from a database, the character generation process ensures two qualities:

1. *Consistency* Elements themselves may carry implications. The forename Lazarus and the title of Duchess are both gendered; by feeding these identifiers into the inference engine and seeing it returns a contradiction, the application knows it can't assign both to the same character.

2. *Uniqueness* No primary element is repeated. Note that this is a narrative choice, not a technical limitation, with the inference engine robust enough to handle generalisations not relevant to *Bad Bohemians*. Indeed, in this particular case, the engine *has* to understand that some identifiers may be non-unique in order to understand gender.

The intermediary information I_i and final information I_f of this cast are also calculated, the former being all information the inference engine can deduce from just a list of primary elements used; the latter, all that will ever be deduced about each character.

Next, clue templates are read from a JSON file: these consist of the clue's unparsed text itself, the subjects alluded to in that text, and the implications carried by each subject (see Figure 1). The inference engine treats each clue as a category, with its subjects the elements thereof. While the player can read 'Lazarus talked to the Earl' and deduce Lazarus is not the Earl, the engine cannot understand raw text - it must be explictely told the sentence has two distinct subjects A and B, that A has forename Lazarus, and that B has title Earl.[2]

Bad Bohemians generates each clue by mapping these subjects onto its cast of 5 characters, parsing in the relevant character information, and checking against its final data for any contradiction. For efficiency, the clue's obviousness is also calculated at this stage (see 2.2). With a space of consistent clues generated, it can now be optimised over...

```
"In one of London's premier dining societies, it is common knowledge that a
#Culprit::Title# #Culprit::Forename# #Culprit::Surname# of  #Culprit::County#
is a swindler, a huckster, and indeed a fraud. Of course, whether the the old boys
on the committee intend to act on these revelations is quite another matter...": {
    "Culprit": [
      [ "Headline", "=>", "THE CONFIDENCE TRICKS OF THE CONFIDENT TRICKSTER" ]
    ]
}
```

Figure 1: An unparsed clue template, a key-value pair of text read by the player and data received by the engine. During processing, hashed information like a possible culprit's name will be written into the clue, and correspondingly added as a forward implication in the data.

[2]It is the responsibility of the narrative designer to ensure the player and engine receive the exact same information by their respective means.

2.2 Optimal Puzzles

But optimise for what, exactly? *Bad Bohemians* defines four metrics for a 'good' Einstein puzzle, given from most to least important.

Consistency No good puzzle contains logical errors. Consistency is treated as a binary quantity: a set of clues either agrees with the final data, or it leads to a contradiction. The application must ensure the former.[3]

Unknowns No good puzzle asks the player to guess elements. *Bad Bohemians*' must minimise the number of primary elements it leaves unmentioned; if its number of unknowns is greater that 0, it will clearly not be solvable.

Ambiguities No good puzzle is underdetermined - that is to say, allows more than one solution.[4] While the underlying optimisation here is very intuitive, reducing possible solutions until only the true one remains, the genetic algorithm will need it formalised.

The inference engine thinks of its information I in terms of element-category pairings: for each element $s \in S$, it creates a subset T' of the elements s could correspond to in category T, and gradually eliminates options from this subset. This allows the definition of the function

$$n_I(s, T) = |T'|,$$

and therefore

$$N_I(S, T) = \sum_{s \in S} n_I(s, T).$$

Einstein puzzles can be thought of as constraint optimisation problems in which the player must minimise $N_I(S^*, T^*)$ for each pair of primary categories S^*, T^*.

Taking I_p to be the information inferred by reading the puzzle's clues alongside I_i, *Bad Bohemians* defines the ambiguities left by a puzzle as

$$\text{Ambiguities}(I_p) = \sum_{s \in S} \sum_{t \in T} \left(N_{I_p}(S^*, T^*) - N_{I_f}(S^*, T^*) \right).$$

Assuming none of the clues contradict the final data I_f, a puzzle will be underdetermined if and only if it has a strictly positive number of ambiguities; reducing to 0 means eliminating, from each subset, all options not present in the one true solution.

Obviousness No good puzzle is *too* easy. The genetic algorithm should provide enough information that the player can arrive at a single answer, but there's no fun in simply stating that answer outright. Creating Einstein puzzles that are *provably* not over-determined is beyond the scope of this application (Vassberg & Vassberg 2009), but it will still optimise puzzles to reduce redundancies.

Let I_c be the information inferred by reading a single clue c alongside I_i, and define the obviousness of that clue by

$$\text{Obviousness}(I_c) = \sum_{s \in S} \sum_{t \in T} \left(N_{I_c}(S^*, T^*) - N_{I_i}(S^*, T^*) \right)$$

[3]There is, of course, a difference between the AI producing accidental contradictions and a writer producing intentionally misleading red herrings...

[4]For context, that's one solution out of $(5!)^4$ (over 200 million!) possible combinations of 25 unique elements across 5 categories into 5 characters.

- the larger this is, the more information can be deduced immediately on reading the clue. The obviousness of a puzzle is taken as the sum of its clues'.

Mark Brown of *Game Maker's Toolkit* describes *Return of the Obra Dinn* as "essentially about cross-referencing information" (2018), where a singular 'eureka!' moment can set off a chain of further realisations. With this in mind, consider the inequality

$$\sum_c \text{Obviousness}(I_c) < \text{Ambiguities}(I_i).$$

The right-hand side represents the initial ambiguity of the puzzle, before the player reads anything; the left-hand side, the sum of each clue's value when read in isolation. If a puzzle has zero ambiguities but satisfies this inequality, then by definition it contains more information than the sum of its parts. Reducing the puzzle's obviousness below Ambiguities(I_i) is exactly what will enable those big eureka moments in *Bad Bohemians* (and the more it can be reduced, the more such moments there will be).

It bears emphasising that defining these metrics is an art, not a science. While the above should shine some light on the process behind this particular project, a different programmer could well come to a different yet equally valid set of priorities.[5]

2.3 The Genetic Algorithm

Though the report has introduced these four metrics, the genetic algorithm will not be concerned with all of them. Pre-processing, after all, only generates clues that guarantee consistency. The following, therefore, sets out a learning AI focused on minimising first unknowns, then ambiguities, and finally obviousness.

At the core of the algorithm is its `ChooseNarrative()` function, which searches for improvements on given puzzle (or, as denoted in the code, a given `narrative`) across a list of potential replacements (`choices`). Crucially, each replacement will need its unknowns, ambiguities and obviousness evaluated to see if it is in fact an improvement, done so in a very specific order due to performance considerations:

1. All `choices` have their unknowns and obviousnesses calculated, a computationally efficient process. Sub-optimal options are immediately discarded: if a puzzle has 0 unknowns and its potential replacement has 1, the function sees no benefit in checking its ambiguities.
2. Each `choice` has their ambiguities calculated, one by one. This necessarily requires use of the inference engine, and cannot be handled during pre-processing, which slows the algorithm down to a crawl.
3. Given this bottleneck, the function immediately returns the first `choice` that improves on the initial puzzle - it is more valuable to stop running the inference engine as soon as possible than search for a better option in the remainder of the list.

The function also comes with a time-out clause, and a maximum number of `choices` it is allowed to check, so that it can best 'cut its losses'.

[5]Consider the following. Because $N_I(S,T)$ only considers two categories at a time, saying 'Lazarus is a Duke or from Hanley Heath' isn't registered as that obvious: it is true that Lazarus has no fewer options in the Titles category, or the County category, but metric fails to account for Lazarus now having far fewer options *in the Cartesian product* of the two categories. It's a subtlety this application is happy to ignore, but that another may not be...

The code of `GeneticAlgorithm.cs` is structured as one would expect. After intialising its first generation from scratch, it will fall into a familiar while loop:

1. *Evaluation.* The fitness of generation n is checked, and the while loop is broken if the best sample cannot be improved upon (or if the algorithm runs out of time).
2. *Selection.* The least fit members of generation n are discarded.
3. *Mutation.* Members of generation n undergo random changes.
4. *Crossover.* Members of generation n cross with one another to spawn generation $n + 1$.

Rather than causing members of the population to change with a low probability of 1% or so, the mutation function actually attempts to mutate every member, every time. The function chooses 2 out of 5 clues in a given puzzle to swap out, then uses `ChooseNarrative()` to search through as many replacements for those clues as possible (i.e. explore 2 out of 5 dimensions of the solution space). Luckily, `ChooseNarrative()` finds improvements infrequently enough that this rather inflexible mutation still feels suitably random!

Notice also that the crossover function is incredibly primitive. Having previously discussed creating puzzles that are more than the sums of their parts, it would follow that it is surprisingly hard to break them down and retain the best elements in isolation. The original model for crossover was to take two puzzles and cross each of their 5 clues into $2^5 = 32$ possible choices, then run these through `ChooseNarrative()` to check if any were viable. Promising in theory, in practice those new puzzles would be consistently worse than their parents, and running them through the inference engine would waste too much time. Far better results emerge when the most optimal sample from generation n is simply copied onto all samples in generation $n + 1$ - to use the evolutionary analogy, this genetic algorithm reproduces by mitosis.

3 Results

In one sense, the results produced by this process are highly subjective. The inherent worth of a puzzle like Figure 2 will vary from player to player: the difficulty, fairness and emergent narratives all lie in the eye of the beholder. Fundamentally, art - and the emotions it ellicits - does not exist to be analysed and optimised in a quantitative fashion.[6]

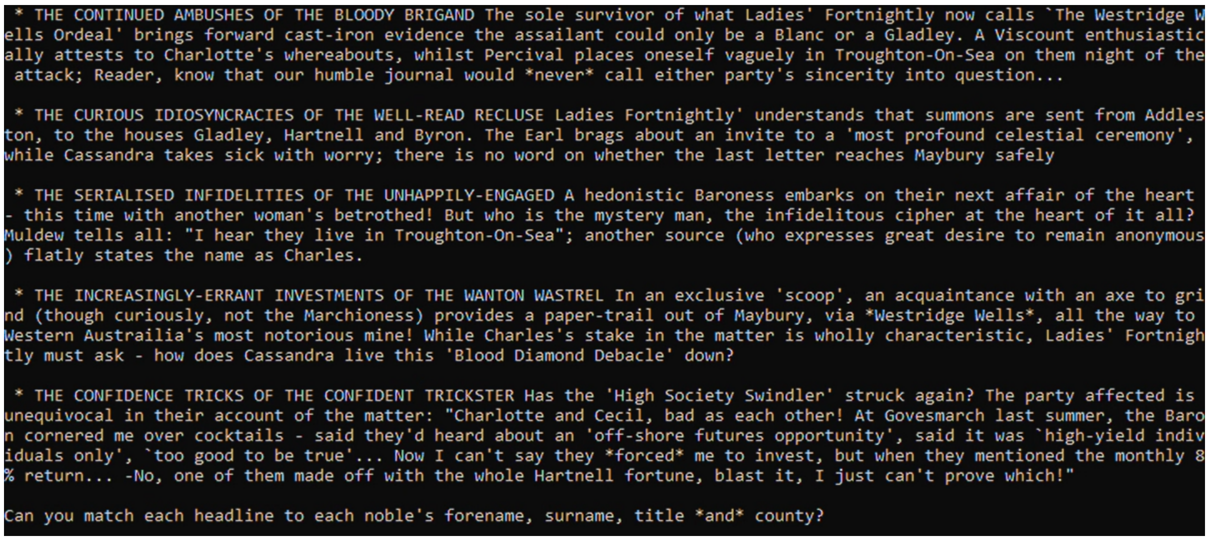
The more nuanced approach would be to evaluate this AI's performance as an extension of the narrative designer. Having essentially outsourced the application's writing to a genetic algorithm, it becomes necessary to check how effectively the output meets the designer's priorities.

In the case of *Bad Bohemians*, these priorities are conveniently quantitive: the algorithm must generate unambiguously solvable puzzles, with not just the solution but the narrative as a whole feeling measurably new each time. Moreover, with contemporary games like *Civilazation VI* (2016) taking only a few minutes to generate a world map that will last for hours, *Bad Bohemians* must produce its results without testing the player's patience. The results of this section, then, exist to provide direction on how to further optimise the program's code.

3.1 Efficiency

As discussed, the genetic algorithm has two possible termination conditions: the puzzle sees no improvement over a fixed number of generations (which this report will call termination via optimisation), or the genetic algorithm has run for a maximum number of seconds (termination via

[6] In this spirit, the reader is encouraged to attempt and evaluate the result in Figure 2 for themself.


```
 * THE CONTINUED AMBUSHES OF THE BLOODY BRIGAND The sole survivor of what Ladies' Fortnightly now calls `The Westridge W
ells Ordeal' brings forward cast-iron evidence the assailant could only be a Blanc or a Gladley. A Viscount enthusiastic
ally attests to Charlotte's whereabouts, whilst Percival places oneself vaguely in Troughton-On-Sea on them night of the
 attack; Reader, know that our humble journal would *never* call either party's sincerity into question...

 * THE CURIOUS IDIOSYNCRACIES OF THE WELL-READ RECLUSE Ladies Fortnightly' understands that summons are sent from Addles
ton, to the houses Gladley, Hartnell and Byron. The Earl brags about an invite to a 'most profound celestial ceremony',
while Cassandra takes sick with worry; there is no word on whether the last letter reaches Maybury safely

 * THE SERIALISED INFIDELITIES OF THE UNHAPPILY-ENGAGED A hedonistic Baroness embarks on their next affair of the heart
- this time with another woman's betrothed! But who is the mystery man, the infidelitous cipher at the heart of it all?
Muldew tells all: "I hear they live in Troughton-On-Sea"; another source (who expresses great desire to remain anonymous
) flatly states the name as Charles.

 * THE INCREASINGLY-ERRANT INVESTMENTS OF THE WANTON WASTREL In an exclusive 'scoop', an acquaintance with an axe to gri
nd (though curiously, not the Marchioness) provides a paper-trail out of Maybury, via *Westridge Wells*, all the way to
Western Australilia's most notorious mine! While Charles's stake in the matter is wholly characteristic, Ladies' Fortnigh
tly must ask - how does Cassandra live this 'Blood Diamond Debacle' down?

 * THE CONFIDENCE TRICKS OF THE CONFIDENT TRICKSTER Has the 'High Society Swindler' struck again? The party affected is
unequivocal in their account of the matter: "Charlotte and Cecil, bad as each other! At Govesmarch last summer, the Baro
n cornered me over cocktails - said they'd heard about an 'off-shore futures opportunity', said it was `high-yield indiv
iduals only', `too good to be true'... Now I can't say they *forced* me to invest, but when they mentioned the monthly 8
% return... -No, one of them made off with the whole Hartnell fortune, blast it, I just can't prove which!"

Can you match each headline to each noble's forename, surname, title *and* county?
```

Figure 2: A puzzle, as generated in the attached video (obviousnesses 186/376).

time-out). The task of the programmer is to set this number of seconds so as to guarantee the best possible puzzles in the shortest possible time.

Figures 3 and 4 provide useful insight in this regard. The underlying data was produced by running the application with `GeneticAlgorithmProcessor.cs`'s time limit fixed at one minute,[7] and `GeneticAlgorithm.cs`'s increased by 30-second intervals (for consistency, `Simulation` class was assigned a constant seed 0, generating the same characters each time).

Points above Figure 3's maximum run-time line (and, due to nuances in the code, points ever-so-slightly below it) can be takem as terminating via time-out; those comfortably below the line can only have terminated via optimisation. As such, the sudden falloff in time-outs after the 240-second mark on the x-axis could suggest allocating more than four minutes to the genetic algorithm produces diminishing returns.

Figure 4 supports this, to some extent. Though it consists of only the $n = 17$ unambiguous samples in the data set, there isn't any suggestion that larger time limits guarentee subtler puzzles. Based on this rather preliminary data, the report is comfortable in allocating a slow-but-not-unreasonably-so 240 seconds to the algorithm; even though the AI is capable of producing a puzzle with obviousness close to 200 in only half a minute, a significantly larger sample would be required to justify reducing the time limit further.

Please note that all results in this section were produced on a mid-range PC. While a 'release' version of *Bad Bohemians*' would work to the 240-second benchmark on a far broader range of devices, the report is satisfied to ignore this consideration at the present, experimental stage of development.

[7]In a more complete analysis, this too would be treated as a variable.

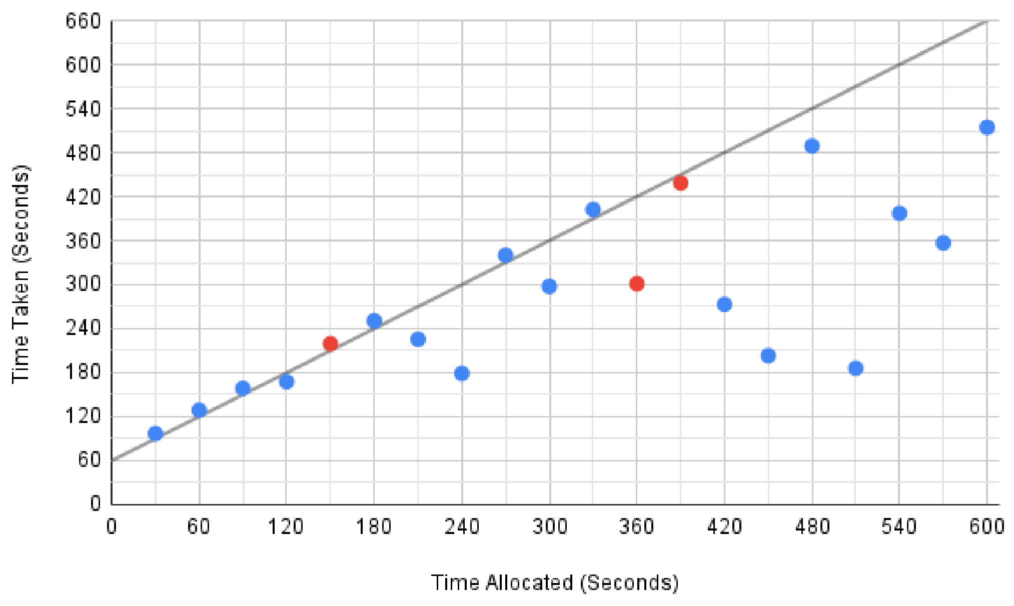


Figure 3: Time allocated to `GeneticAlgorithm.cs` vs. time taken to generate puzzles (blue/red points representing unambiguous/ambiguous puzzles, respectively; black line representing a rough maximum run-time, the sum of `GeneticAlgorithm.cs`'s and `GeneticAlgorithmProcessor.cs`'s time limits).

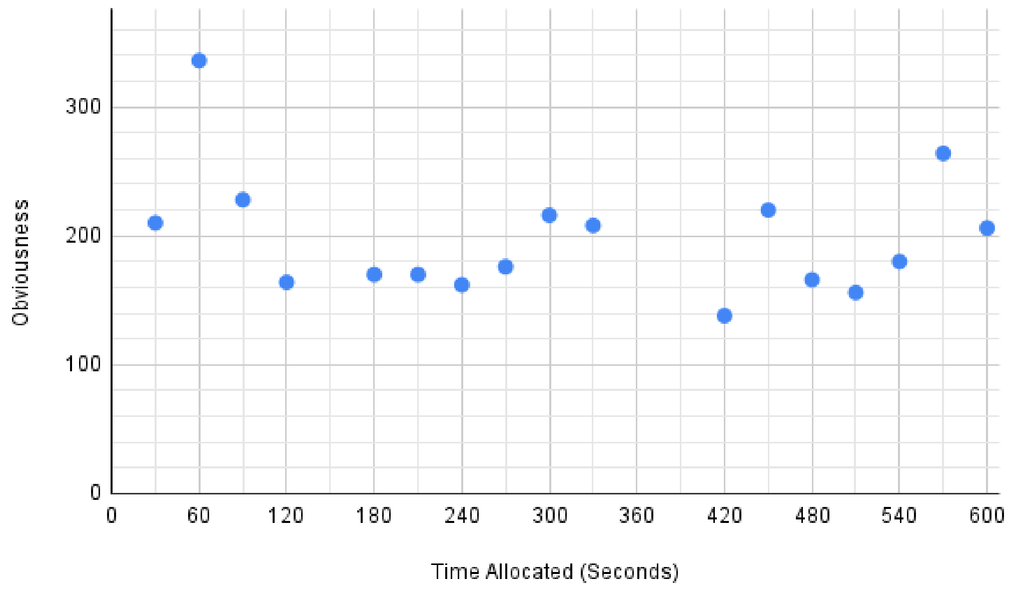


Figure 4: Time allocated to `GeneticAlgorithm.cs` vs. obviousness of generated puzzle.

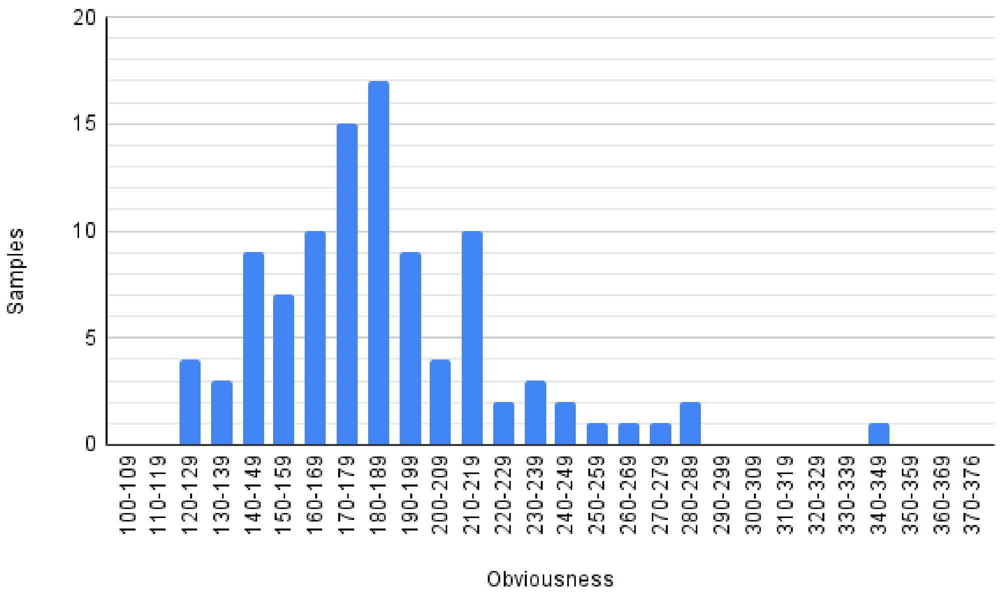


Figure 5: Sampling distribution of obviousness across $n = 105$ puzzles ($\bar{X} = 183.75$, $S = 38.04$).

3.2 Replicability

With a closer look at Figure 3, though, there is a noticeable cause for concern. At the 360-second mark, the genetic algorithm has returned an ambiguous puzzle, but clearly terminated via optimisation - in other words, the algorithm had the time to make improvements *but chose not to.* To better understand how reliably the AI produces 'good' puzzles will, accordingly, require a more reliable set of data.

For the purposes of this report, the genetic algorithm is run $n = 120$ times (with `GeneticAlgorithmProcessor.cs` allocated 60 seconds, `GeneticAlgorithm.cs` allocated 240 second, and the seed again fixed at 0). Two key results emerge:

- Of the 120 puzzles generated, 15 contain ambiguities (12.5%).
- Of the 105 unambiguous puzzles, the most optimal has an obviousness of 124, out of a possible $\text{Ambiguity}(I_i) = 376$; the mean obviousness is 183.75 (see Figure 5).

At a glance, this would appear extremely disappointing: even assuming the value of 124 reached is the seed's true, global minimum for obviousness, the AI cannot replicate the result in any sort of consistent fashion.

Figure 5 must therefore be understood in the appropriate context. It is a common criticism of genetic algorithms that they will often get stuck at local optima; for *Bad Bohemians*' purposes, this is not a bug but a feature. While players will not receive the same seed twice per se, clue templates are conditional on very few categories, leaving many sets of characters 'isomorphic' to one another.[8] Even under equivalent starting conditions, the genetic algorithm can maintain variety by optimising towards distinct (yet consistently low!) local minima.

The 12.5% of puzzles left ambiguous are far more concerning. Since each scandal has one trivial

[8]For example: given the ingrained gender roles of the time a clue template may require that the highway robber is a man - but it far less likely to care whether they are called Lazarus or Percival.

template that explicitely states the corresponding character's full identity (e.g. 'Duke Lazarus Sallow of Hanley Heath is the highway robber'), don't the AI's priorities dictate that it should be returning completely trivial puzzles with 0 unknowns and 0 ambiguities instead? Unfortunately, it follows from the methodology of 2.3 that if the genetic algorithm has reduced a puzzle's unknowns to zero, it simply isn't allowed to swap in trivial clues if it would increase unknowns in the process. Possible fixes for this are discussed as part of 'Conclusions'.

3.3 Variety

While *Bad Bohemians* has a not-insubstantial word count of ≈ 1500, this translates to only 30 clues, each specific to one or two of the five stock characters. The report here considers the frequencies of each clue across $n = 10$ samples, one for each seed 0-9 - bearing in mind that the genetic algorithm has been designed to support different writers with different stories, this is by far the most case-specific set of results provided.

'Robber' Clues	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**
Uses	0	5	1	3	1	0

'Con Artist' Clues	**#7**	**#8**	**#9**	**#10**	**#11**	**#12**	**#29**
Uses	0	0	3	4	1	1	1

'Recluse' Clues	**#13**	**#14**	**#15**	**#16**	**#17**	**#30**
Uses	0	0	0	0	10	0

'Adulterer' Clues	**#18**	**#19**	**#20**	**#21**	**#22**	**#23**	**#29**
Uses	0	0	1	7	1	1	0

'Investor' Clues	**#24**	**#25**	**#26**	**#27**	**#28**	**#30**
Uses	0	2	2	2	4	0

Figure 6: Frequencies of each clue in *Bad Bohemians*, across $n = 10$ seeds.

Lacking statistical significance, this data is best understood as a tool for the narrative designer, collected as form of progress check. From Figure 6, they might conclude that the investor currently sees a lot of variety, and instead focus on producing content for the adulterer or the recluse over their next writing sprint. It can also highlight how highly situational certain clues are: clue #8 requires at least 3 men, one of whom is the con artist; clues #1, #7, #13, #18, #24 are the afore-mentioned trivial clues, which absolutely *shouldn't* be used except as a last resort.

4 Conclusions

So, where does this leave the AI at the heart of *Bad Bohemians*? Do its limitations lie in a distinctly finite set of clues, an inefficient inference engine, or its own underlying design? Before trying to identify the improvements to be made, it's important to emphasise the most fundamental achievement of this whole project: I have created an algorithm capable of using *my own clues* and *my own logic* to generate puzzles that challenge *me*. From a personal perspective, this alone qualifies the AI as a success.

It is perhaps this first-hand understanding of what the algorithm is capable of, though, that makes the inconsistencies so frustrating. As set out in 3.2, 87.5% of puzzles generated being unambiguous is by no means a bad success rate,[9] but it clearly needs improvement. For all the 'soft' design solutions here - accepting *any* valid answer from the player, or even just leaving the last step of the puzzle to trial-and-error - none are satisfying from a mathematical perspective. I may need to re-evaluate my underlying methodology from first principles.

Speaking of re-evaluation, the inference engine cannot go unmentioned: creating it has simulataneously been the most gratifying and most frustrating part of the entire assignment. To some extent, the challenge came from trying to write its logical rules as generally as possible, not yet knowing what the final application would look like or what functionality it would need. Now, with a better understanding of the scope of *Bad Bohemians*, it should be possible to pare the engine back, sacrificing generality for efficiency. The genetic algorithm would certainly benefit - having been designed to avoid using the rules-based system where possible due to the poor performance, being able to run it more frequently could allow for larger populations, more exciting mutations, even the introduction of a non-trivial crossover function as I develop the project further.

And make no mistake: *I am going to develop it further.* To that extent, seemingly-lateral features could well provide solutions for (or at least, better diagnoses of) the genetic algorithm's current limitations. Take, for instance, the text generation: *Bad Bohemians* currently produces heavily authored narratives, largely similar save for the different names swapped in each time. It seemed like building a working text generator would be just too irrelevant to what was being assessed, but on reflection this would have been of great help to the AI. Better text does not just mean a more artistically-fulfilling narrative, it means a bigger solution space for the genetic algorithm to optimise over, and hence more options for reducing ambiguities without increasing unknowns at a given moment. This feature may be just what 3.2 requires!

For now, *Bad Bohemians* is best understood as a proof-of-concept. Given all the optimisations and oversights to be made, it's worth understanding that this project (at least to some extent) always had the deck stacked against it. 1983's *Murder on the Zinderneuf*, subject of *The Digitial Antiquarian*'s afore-mentioned restrospective, is a bold commercial attempt at the procedural detective game panned by Maher; Sheldon Klein's murder-mystery generator (1973) is regarded as one of "academic story generation's most famous failures" (Ryan 2018). I'm proud to present a genetic algorithm that exists as part of that deeply experimental and deeply over-ambitious lineage - even if it is yet to fully crack the case of the procedural whodunnit.

References

Color Gray Games (2022), *The Case of the Golden Idol*, Playstack.

Compton, K. (2019), 'Getting Started with Generators', *in* T. Short & T. Adams, eds, '*Procedural Storytelling in Game Design*', 2nd edn, Boca Raton, FL, USA: Taylor & Francis, pp. 3–17.

Firaxis Games (2016), *Sid Meier's Civilization VI*, 2K Games.

Free Fall Associates (1983), *Murder on the Zinderneuf*, Electronic Arts.

Game Maker's Toolkit (2017), 'What Makes a Good Detective Game?', (online video) Available at: `https://www.youtube.com/watch?v=gwV_mA2cv_0`. (Accessed: 11 January 2022).

[9]Especially with the worst number of ambiguities observed being 34/376. The definition set out in 2.2 does not express a linear relationship between the proportion of ambiguities and number of possible solutions; an undetermined puzzle with 34/376 ambiguities will only allow for a few solutions out of the afore-mentioned $(5!)^4$.

Game Maker's Toolkit (2018), 'How Return of the Obra Dinn Turns You Into a Detective', (online video) Available at: `https://www.youtube.com/watch?v=V0qxLrFycrc`. (Accessed: 11 January 2022).

Gibson, P. & Byrne, J. (1991), 'NEUROGEN, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks', *in* '*1991 Second International Conference on Artificial Neural Networks*', Bournemouth, UK: 18-20 November.

Haubenwallner, K., Seidel, H. & Steinberger, M. (2017), 'ShapeGenetics: Using Genetic Algorithms for Procedural Modeling', *Computer Graphics Forum* **36**, 213–223.

Holland, J. (1992), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 1^{st} edn, Cambridge, MA, USA: MIT Press.

Klein, S. *et al* (1973), Automatic Novel Writing: A Status Report, Technical report, University of Wisconsin, Madison.

Maher, J. (2013), 'Free Fall, Part 2: Murder on the Zinderneuf', Available at: `https://www.filfre.net/2013/02/free-fall-part-2-murder-on-the-zinderneuf/`. (Accessed: 8 January 2023).

Pope, L. (2018), *Return of the Obra Dinn*, 3909.

Ryan, J. (2018), Curating Simulated Storyworlds, PhD thesis, University of California, Santa Cruz.

Vassberg, D. & Vassberg, J. (2009), 'Is Einstein's Puzzle Over-Specified?', *in* '*21st Century Challenges in Computational Engineering and Science*', Princeton University, NJ, USA: November.