Measuring Difficulty of Novel Clockwork Puzzle Instances Using Evolutionary Algorithms

Connor Gregor

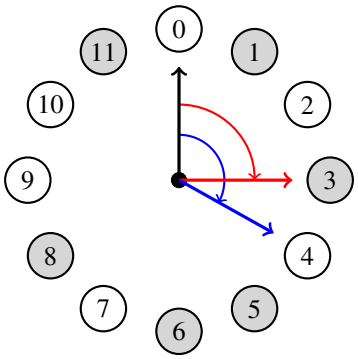

Fig. 1. Initial State of Simple Clockwork Puzzle

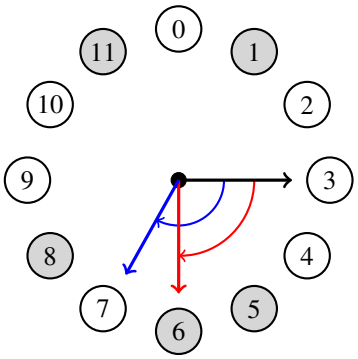

Fig. 2. Updated State of Simple Clockwork Puzzle

***Abstract*—This study debuts the clockwork problem a novel type of combinatorial puzzle; the rules and general variations of the puzzle are explained in order to showcase the puzzle's interesting properties and motivate interest in the puzzle. Following this, the "Do What's Possible" representation for an evolutionary algorithm is used to solve instances of the clockwork puzzle. The results of the evolutionary algorithm across many instances are analyzed to create a difficulty measure for said instances. This type of computational intelligence is preferred as the evolutionary algorithm adequately mimics the approach of a human player who lacks prior knowledge about the puzzle.**

I. Introduction

From a subjective viewpoint, the most interesting puzzles can be presented to a player while providing minimal instruction. The player must make their initial moves and develop their understanding of what is possible while contained within the puzzle's structure. Puzzles of this nature usually start with simple instances, then pit the player against a "difficulty curve" of successively more challenging puzzle instances.

Early video games serve as a great example for this type of puzzle; minimal, if not zero, instruction is given to the player as they are left to beat the game on their own. The clockwork puzzle introduced by this paper serves as another example for this type of puzzle.

Example 1 serves as an introduction to the clockwork puzzle's premise.

Connor Gregor is with with the Department of Mathematics and Statistics at the University of Guelph, {cgregor}@uoguelph.ca

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

Example 1. *You are presented with an object similar to an analogue wall clock which has a clock hand pointing to the number zero. The numbers of the clock are illuminated in sequence by lights as shown in Fig. 1.*

You are presented with two options: to rotate the hand clock-wise by either three or four increments. When the hand is moved, the number that the hand lands upon has its light toggled between the settings "on" and "off". Fig. 2 demonstrates the new clock state made after using the "rotate by three increments" move. Using these two options in some sequence, can one create a state where every light is illuminated?

In general, a clockwork puzzle can be thought of as a single-player combinatorial game where one uses a designated action set in order to transform an initial state into a goal state. As inherent properties: any move made by the player can be undone and if a puzzle instance is deemed solvable, then it is impossible for the player to negate this quality.

This "always solvable" quality is shared by puzzles like the Rubiks Cube [1] and the Tower of Hanoi [2] where a player can always reverse what they have done as they attempt to construct a goal state. Reversal of clockwork puzzle moves tends to be a fare more lengthy process for the player than it is in those previous two examples. This puzzle quality is contrasted by simple puzzles like the self-avoiding walk [3]. Puzzles of this variety can trap the player into unsolvable situations should they make a poor choice of moves.

The specific clockwork puzzle described in Example 1 is one of its simpler instances; many other instances can be made through control of certain parameters specific to the

puzzle. To be able to categorize the various clockwork puzzle instances by difficulty would be a useful feature for puzzle designers who wish to incorporate the clockwork puzzle in a game. In order to do this, computational intelligence techniques can be applied on a curated sample of puzzle instances. Solutions and run times provided by the algorithm can then be analyzed in order to determine the difficulty rating of an instance.

With this goal in mind, the layout of this paper is as follows. Section II provides a thorough description of the clockwork puzzle and outlines all the parameters required to design a specific instance. Section III then discusses the "Do What's Possible" (DWP) representation [4]: an evolutionary algorithm (EA) which creates solutions for clockwork puzzle instances. Section IV explains interesting puzzle instances and the parameters used by the algorithm. Section V lists off the instances examined and summarizes the results obtained on the various instances in tabular form. Section VI states general conclusions about instance difficulty that can be made based on the results; closing remarks then mention future work intended for the clockwork puzzle.

II. Clockwork Puzzle Structure and Rules

An instance of the clockwork puzzle can be described by five parameters. The first relevant parameter is the number of lights n. Starting from zero, the values of $\mathbb{Z}_n$ are arranged clock-wise in increasing order so that the lights 0 and $n-1$ are adjacent

The second relevant parameter for a clockwork puzzle instance is the number of lights settings c. Referring back to example 1 again, each light was able to have one of two settings: "on" or "off". Every time that the hand of the clock landed upon a number, its associated light would swap settings. For clockwork instances with c light settings, the light settings are arranged in a fixed cycle. The order of settings in the cycle dictates which settings are "adjacent" to one another. The best way to express this cycle is to have each light of the clockwork puzzle be represented by a value of $\mathbb{Z}_c$. Each possible light setting maps onto exactly one value of this set.

Visually, clockwork puzzle instances with more than two light settings is presented best by using a spectrum of colors arranged in a specific order. Examples of this would be the colors of a traffic light for $c = 3$ or a rainbow for $c = 7$.

The third parameter required in order to properly describe a clockwork puzzle instance is its initial state, w_0. Prior to describing an initial state of a clockwork puzzle, the defining of a clockwork puzzle state is warranted.

Definition 1. *A state, w, of a clockwork puzzle instance with n lights and c light settings is any combination of ordered lights in conjunction with a specified hand position. The name for the set of states of the clockwork puzzle is declared to be $W_{n,c}$.*

Lemma 1. *For a clockwork puzzle, the set $W_{n,c}$ contains nc^n unique states. Equivalently, the set $W_{n,c}$ can be thought of as $\mathbb{Z}_n \otimes (\mathbb{Z}_c)^n$.*

Proof. Each of the n lights must be set to one of c settings. These n choices can be made independently from one another thereby creating at least c^n different possible states. The hand position on the clock can be placed upon one of the n lights; since this is another independent decision, the number of states will equal nc^n.

The other portion of the proof for this lemma comes from computing the size of $\mathbb{Z}_n \otimes (\mathbb{Z}_c)^n$ while noting that each set in the cross product is equivalent to one of the aforementioned $n+1$ independent choices. □

With this understanding of the set of states as described in Lemma 1, a numerical representation can be used to describe any state of the clockwork puzzle. This numerical representation uses a sequence of $n+1$ numbers that list off the position of the hand followed by the settings of the n lights in sequence. For the sake of clarity, a comma should separate the hand position value from the light setting values. The entry before the comma is the k-component of the state, while the string of values following it is the L-component of the state. As an example, the clock state in Fig. 1 can be written as $(0, 010101101001)$ if one asserts that lights being "white/on" or "grey/off" correspond to the numbers zero and one respectively; similarly, the state in Fig. 2 is written as $(3, 010001101001)$.

Now that the states of the clockwork puzzle have been properly defined: the initial state, w_0. is simply a single state from $W_{n,c}$ that is presented at the start of a puzzle instance.

The fourth relevant parameter of a clockwork puzzle instance is its goal states, $W_G \subset W_{n,c}$. The goal states are a collection of states that the player strives to construct. In Example 1, the goal states were thought of as the states with L-components of all zeroes (all lights turned on); this created n different possible goal states. In a general puzzle instance, the condition of similar L-components can be disregarded. Any collection of states in $W_{n,c}$ can be designated as the goal states of the puzzle instance; this study avoids dealing with such general goal state collections and focuses specifically on goal state collections as seen in Example1.

The last parameter used to specify a clockwork puzzle instance is the action set A. The action set describes the available clock hand motions that the player may use to solve the puzzle; the hand motions in Example 1 serve as a very simple action set.

In general, action sets are made up of actions, a, that alter states. The use of an action on a state of the clockwork puzzle can be thought of as two states being combined together by a special type of operator. Prior to introducing this operator, let σ be the permutation which cycles through ordered elements using the following bijective mapping:

$$\sigma L = \sigma(l_0, l_1, \ldots, l_{n-1}) = (l_{n-1}, l_0, l_1, \ldots, l_{n-2}).$$

Definition 2. *Given two clockwork states $w_1 = (k_1, L_1)$ and $w_2 = (k_2, L_2) \in W_{n,c}$, define **Modulo Rotary Addition** of these elements as:*

$$w_1 +_r w_2 = \big((k_1 + k_2) \pmod n, (\sigma^{k_2} L_1 + L_2) \pmod c\big).$$

In other words,

- *Treat L_1 as a set of ordered objects and rotate it by the permutation σ^{k_2} (indices shift by k_2) to generate M.*
- *Perform the operations $k_1 + k_2 \pmod n$ and $M + L_2 \pmod c$ to obtain the two components of $w_1 +_r w_2$.*

Applying an action a onto a state w_i to create a new state w_{i+1} is done by the equation: $w' = a +_r w$. Though any action can be described by a state, it turns out that an action can be described more intuitively in the encoded form of $a = (k, \sigma^k L)$.

The k-component of an action describes the number of increments the current state's clock hand ought be rotated clock-wise. Meanwhile, The L-component of an action describes how lights ought to be toggled relative to the new clock hand position; the value of l_0 states what modification ought to be made to the light that the hand lands upon while l_1 states what would happen to the next light in the sequence and so on with l_2. By using σ^k to describe a, similar actions can be grouped together based on how they alter lights relative to hand position.

Several unique actions are what make up the action set A. The action set is constructed by considering each possible action (state) and deciding whether or not to include it in the set; thus, there are 2^{nc^n} different possible action sets.

To relate actions back to Example 1: the action set in this case held two elements; by using the aforementioned notation, the actions can be expressed as $(4, \sigma^4 e_0)$ and $(3, \sigma^3 e_0)$ where e_0 is the first standard basis vector; for the sake of neat notation, linear combinations of the standard basis vectors e_i will almost exclusively be used to describe the L-components of actions hereafter.

Using these five parameters, one can create an instance of the clockwork puzzle, but knowing how to set up the puzzle by itself may not develop a deep enough understanding. In order to aid with this possible issue: a second more complicated clockwork puzzle ought to be presented so that one can better grasp how the game functions.

Example 2. *Consider a clockwork instance on six lights. Each light can take on one of four possible settings arranged in the following cycle: red, yellow, green, blue, then back to red. The cycled colors are mapped respectively onto the values of $\mathbb{Z}_4$ starting from zero. The initial state of the instance is provided by the lop left state in Fig. 3.*

The action set A for this instance is $\{(3, \sigma^3(e_1 + e_5)), (1, \sigma(3e_0 + e_3), (5, \sigma^5(3e_0 + 2e_1 + e_2))\}$. The implementations of the three actions are shown in the top right, bottom left, and bottom right pictures respectively.

With the concept of clockwork puzzle instances now described in full, the idea of a difficult one becomes more clear. The choice of $W_{n,c}$ based on the parameters n and c, the choice of initial state $w_0 \in W_{n,c}$ as well as the collection of goal states $W_G \subset W_{n,c}$, and assignment of the action set $A \subset W_{n,c}$ at the player's disposal; this combination of factors can greatly alter a player's perceived difficulty of the puzzle. The choice of these parameters is important because

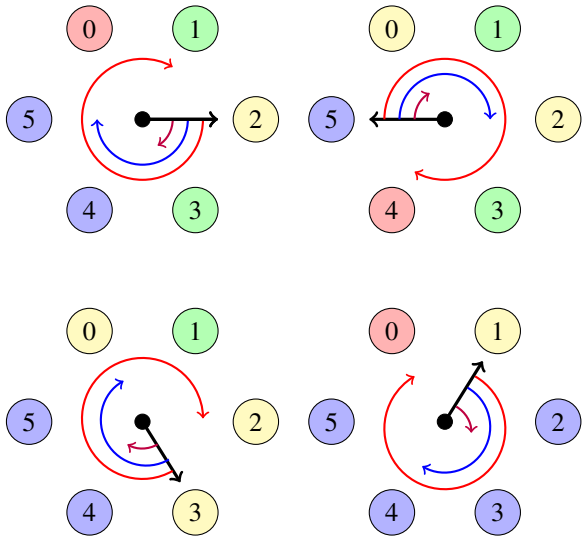

Fig. 3. Clockwork Puzzle States After Various Actions

sloppily arranged instances can end up being impossible. This impossibility lies in the action set being unable to create any of the goal states when starting from the initial state of the instance; further discussion on solvable instances is reserved for Section IV.

III. The Do What's Possible Representation

As mentioned, instances of the clockwork puzzle can be solved using the DWP representation. DWP is an EA that uses a generative representation; generative representations do not directly evolve solutions to a problem. Instead, generative representations evolve blueprints that can be decoded into solutions to a problem.

The DWP representation does this by evolving an *alternator ring*, which is read by a decoder in order to create solutions to a problem. An alternator ring for the clockwork puzzle can be encoded by placing one of $R + |A|$ symbols into one of R positions on the ring; these symbols are then read off cyclically in order to create a solution to the clockwork problem. The first R of these symbols are known as the *alternators* [5]; Each of the R symbols are written as some value between one and R. When read by the DWP representation, an alternator will generate one of two outcomes depending on the previous outcome of said alternator.

1) If this is the first time that the alternator has been read or if it previously followed the second instruction for this operator: read the value of the alternator and jump to that position in the ring.
2) If one previously followed the first instruction for this operator: ignore the value of this alternator and continue reading symbols in the sequence.

The remaining $|A|$ symbols that can be encoded into the alternator ring are meant to correspond to a move from the action set of the clockwork puzzle. Upon reading one of these symbols, the action corresponding to the symbol is

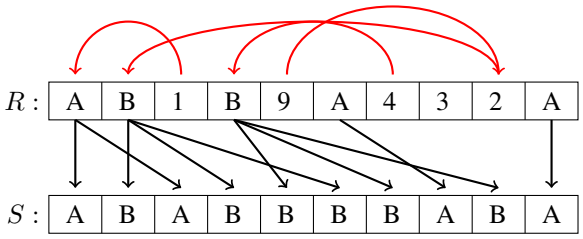

Fig. 4. Example of String Generated by Alternator Ring

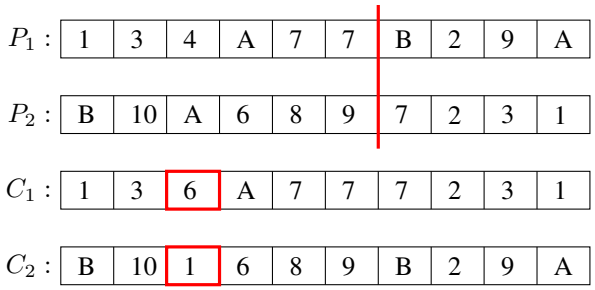

Fig. 5. Crossover Between Two Parents and Mutation of Children

implemented on the current state of the clockwork puzzle instance. In order to avoid reading an alternator ring indefinitely, a read limit of L symbols is imposed on the algorithm.

By continually reading from the alternator ring, a string of actions that attempts to solve the clockwork puzzle is formed. By its design, the alternator ring of the DWP representation can create strings of actions which are highly patterned; through a clever use of alternators this pattern becomes long and convoluted.

Fig 4 gives an example of an alternator ring being read. The first two entries copy their values onto the string; the third entry moves the ring reader back to the first entry of the ring where the same two entries are read again. As it moves to the third entry a second time, the alternator is skipped and the ring reader moves to the fourth entry which copies its value onto the string. The remainder of the string is formed by following similar logic.

There are a number of reasons why the DWP representation is an excellent choice of algorithm for solving clockwork puzzles. First of all, the required length for a string of actions that solves an instance of the clockwork puzzle is not known, or fixed, in general; if one were to use another type of EA, many approaches force one to evolve solutions of a fixed length. These solutions would either be so small in length so that they don't encode enough actions to make a goal state or so long in length that evolving them ends up being a computationally lengthy and wasteful process. Previous work by Ashlock [6] has shown the DWP representation achieving superior results to direct representation on problems such as the gray codes and the self-avoiding walk when evolving alternator rings of a length far shorter than that of a solution evolved by the direct representation; this result supports the notion that DWP may not only be effective at solving the clockwork puzzle, but may also do so in a more efficient fashion.

When using an EA with the DWP representation on the clockwork puzzle: a population of P alternator rings are initially generated at random; the random generation fills each ring entry by placing one of the $R + |A|$ values based on a uniform distribution. Each ring is assigned a fitness value based on the specific fitness function for the clockwork puzzle.

This fitness function assigns a value to a ring by looking at the set of states made from applying the action sequence generated by the alternator ring's instructions onto the initial state. For each state that is created by reading off an action, the individual fitness of a state, w, is computed using the following formula:

$$fitness(w) = \sum_{i=0}^{n-1} (c - l_i) \pmod{c}$$

After applying this fitness function to each state, the ring's fitness is the minimum fitness obtained by any single state in the set it generated. The formula written for the fitness function assumes that the goal states of the clockwork instance are those with L-components of all zero. By plugging in one of these goal states, the returned fitness is zero, thus the goal for a ring is for it to create a state with the lowest fitness possible.

Once each ring in the population is assigned a fitness value, the algorithm evolves said population. Gradual improvements are made by performing one-point crossover [7] on the population and mutating when appropriate. One-point crossover is used specifically because alternator rings have a fragile structure as each entry of the ring has its usefulness supported by the ring's other entries. To avoid wrecking promising rings: gradual change is the preferred approach.

Crossover is done by performing tournament selection [8] wherein a subset of the population of size T is selected at random. The two most fit and least fit members of the tournament are identified. The two most fit rings from the tournament are temporarily labeled as parents; these parent rings are then combined to create two new child rings who will usurp the positions of the two least fit population members from the tournament.

Crossover creates these children by choosing a specific entry j along the ring length, R. Each child copies the ring entries of one parent up to and including its j^{th} entry while copying the other parent for all remaining entries; the difference between the two children comes from the initial choice of parent to copy. In order to maintain a diverse population, mutation is also applied by selecting random entries of each newly created child ring and overwriting them. The new children have their fitness evaluated, replace the unfit tournament entrants in the population, and await future tournament selections. Fig. 5 demonstrates this explicitly with two parents who perform crossover at their sixth entry; the children then have their third entry mutated.

The EA is expected to do N instances of tournament selection. Should a ring with zero fitness be obtained by the

algorithm, then the run is halted and the number of crossover events done prior to solving the clockwork puzzle instance is recorded; such a run is viewed as a success. Along with the number of crossover events, the algorithm also records the length of the action string made by Fthe ring which solved the puzzle instance. The specific choices for L, R, T, P, and N act as hyper-parameters for the EA. Reasons behind specific selections for each parameter will be discussed more in Section IV.

One may ponder why an EA is being used rather than a best-first search algorithm [9]. The best-first search precisely finds the shortest possible string of actions to solve a clockwork puzzle instance; however, instances with short or long solutions don't necessarily conflate easy or difficult clockwork puzzle instances. Furthermore, the best-first search algorithm considers many states at once while looking for a solution; once an approach proves fruitless, the algorithm looks back to consider an earlier state from the puzzle instance. Such an approach would not be valid under the clockwork puzzle rules as each move is meant to carry lingering consequences; thus, the EA is used instead.

The fashion in which the DWP representation attempts to solve clockwork puzzle instances agrees with the rules properly. Each move made cannot be reversed easily and instances which can be solved through the use of simple patterns are quickly identified due to DWP's affinity to such structures. For these reasons, is also believed that this representation better impersonates the approach that a human player may take when solving a clockwork puzzle instance. Thus, instances which are identified as difficult by the EA may be diagnosed similarly by a human player.

IV. Experimental Design

Now that the clockwork puzzle and the EA have both been introduced, the experiment of this paper can be presented more clearly: several instances of the clockwork puzzle are solved the EA in order to discern a measure of difficulty for said instances. These instances are also presented to a panel of human players who qualitatively assign difficulty to them as well.

Prior to describing the format for the analysis of difficulty, reasoning in regards to the choice of clockwork puzzle instances provided to the EA requires mention. Consider that many clockwork puzzle instances are unsolvable; it is trivial to discern why some instances are unsolvable, but others cleverly disguise their unsolvability.

Example 3. *For the clockwork puzzle instance shown in Fig. 6, The initial state belongs to $W_{6,2}$, any state whose L-component entries all share the same value is a goal state and the action set permitted is $A = \{(1, \sigma(e_0)), (4, \sigma^4(e_0))\}$.*

No matter what string of actions one uses, it is impossible to create a goal state.

Adjacent to the conduct of this study, rigorous mathematical work has developed a set of properties that guarantee a clockwork puzzle instance is solvable, thus avoiding the annoyance of unsolvable instances. This is done by examining

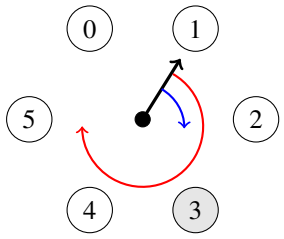

Fig. 6. An Unsolvable Clockwork Puzzle Instance

the action set and initial state of an instance with states in $W_{n,c}$ while presuming the goal states are a collection which all share the same L-component.

At this time, the properties that assert solvability require the constraint that the L-components of the action set are either all zeroes or the first standard basis vector e_0. These types of actions correspond to moves that don't alter lights and moves that alter the light which the clock hand lands upon respectively. To avoid "redundant" or "boring" instances, it is assumed that the k-components of actions do not collectively share a greatest common divisor – other than one – with n; it is also assumed that $|A| > 1$.

Using only these two constraints the instance of a clockwork puzzle, one constructs the set D. The elements of this set are n, the k-components of the actions with L-components of zero, and the differences between the k components of all the actions that have L-components of e_0. Calculate the greatest common divisor among all elements in D and label it as Q. If $Q = 1$ then regardless of the value of c, any initial state can become any goal state; as $Q > 2$ or $c > 2$, unsolvable instances become possible, but one can ensure a solvable instance by carefully selecting w_0 relative to W_G.

The detailed explanation on how Q guarantees solvability requires a generous amount of devoted space alongside supplemental definitions and can be provided on request. To help develop a quick understanding: Q acts as a value that specifies what subset of the nc^n states are reachable from an initial state w_0. The value of Q implies that there exist $\frac{n}{Q}$ reachable states with the same L-component as an arbitrary reachable state w; from these states, only $\frac{n}{Q}$ states with unique L-components can be made using the action set. The result follows after additional elaboration.

Once an instance is solvable, it is sensible to assume that supplementing its action set with additional moves would not compromise solvability and only introduce greater versatility to the player as they solve the clockwork puzzle instance. With solvability in mind, the instances which will be experimented upon by the EA will attempt to diversify across different settings while still ensuring solvability. To alter each setting individually allows for insight to be obtained on what factors contribute towards the making of a difficult puzzle.

These settings include the alteration of four of the five parameters for a clockwork puzzle instance. The number of lights on the puzzle, the number of settings available to a light, the initial state of the puzzle, and most importantly of

all: the action set made available to the puzzle.

The goal states of the experimented instances are to be fixed as the collection of states with L-components of all zeroes; this has been done in order to streamline the EA's fitness function. Putting aside efficiency, a computer would not appreciate the difficulty of having to create an irregular goal instance since the visualization of a state means nothing to it. Two instances can have identical action string solutions while only differing in their choice of w_0 and W_G. Previous work done by Kotovsky [10] on the Tower of Hanoi puzzle has shown that two puzzle instances can be harder for intelligent human players even when they have identical action string solutions. Such visual confusion is lost on an EA and so is eschewed for this study.

Along with the parameters of the clockwork puzzle, the EA parameters have also been calibrated for testing clockwork puzzle instances. The length of entries for the alternator rings and the ring reading limit were set to $R = 15$ and $L = 200$ respectively. The population of rings being evolved was set to $P = 28$ and the number of crossover iterations permitted prior to abandoning a run was set at twenty-thousand.

Using these hyper-parameters, twenty runs of each puzzle instance were attempted. As mentioned briefly at the end of Section III, the EA recorded two pieces of information from each run of a clockwork puzzle instance: the number of crossover events required to create a ring that generates a goal state and the length of the string of actions required by said ring to make a goal state.

These two values have been recorded for every run in each instance. The two results are averaged across all twenty runs for each clockwork puzzle instance and a ninety-five percent confidence interval has been constructed as well. Using these values it is reasoned that the more iterations that it takes the EA to solve a clockwork puzzle instance: the more difficult this instance must be for a human player.

To help support this view, the action string lengths constructed by the EA can be used as supporting evidence to this claim; puzzle instances whose solutions have longer action strings can be thought of as more difficult. This is due to the DWP representation relying upon trial and error as it gradually hones in on a goal state. It does so by gradually sculpting elaborate action strings. In general, a long action string does not imply a puzzle instance is difficult as the string may be highly patterned and obtained almost immediately by DWP.

Meanwhile, the panel of human players are presented the suite of clockwork puzzle instances and prompted to solve said instances. The players are encouraged to rank the puzzles in terms of difficulty to the best of their ability. Comparisons will then be drawn between the qualitative difficulty experienced by the players and the quantitative difficulty found by the EA.

V. Results

The first set of clockwork puzzle instances focused demonstrating the variable difficulty that comes from altering the initial state of the instance. In order to do so, each instance uses the set of states $W_{12,2}$ and the action set $A = \{(7, \sigma^7 e_0), (5, \sigma^5 e_0), (3, 0)\}$. The clockwork puzzle instances in this first set were made distinct by using five distinct initial states. The k-components were all set to zero while each individual instance had the following L-components:

- *(A1)* All lights turned off.
- *(A2)* Every even-numbered lights begin turned on.
- *(A3)* Every numbered light that can be evenly divided by three begin turned on.
- *(A4)* Light at sixth position begins turned on.
- *(A5)* "Noisy" mix of light settings.

The term "noisy" used for instance *(A5)* is meant to suggest that light settings were done at random with the hope of creating a sequence of lights entirely lacking in pattern. Table I shows the results for each instance; preliminary observation of the results suggests that while the EA favored certain instances, it found solutions to all of them quite quickly. The human panel of players found that they experienced a similar hierarchy of difficulty as the noisy and single toggled light instances proved to be more difficult then the other three; players also felt that all instances in this first set were easy.

	Iter.	Iter. 95% CI	Length	Length 95% CI
A1	0.5	±0.6	17.7	±7.1
A2	24.3	±9.6	45.6	±13.5
A3	42.1	±17.1	48.8	13.3
A4	203.2	±124.0	33.3	±11.1
A5	47.1	±21.6	37.4	±10.5

TABLE I
VARIOUS INITIAL STATES FOR PUZZLE INSTANCES

	Iter.	Iter. 95% CI	Length	Length 95% CI
B1	0	±0	12	±0
B2	52.1	±18.2	55.8	±10.4
B3	69.3	±22.3	± 55.4	±9.6
B4	722.5	±334.4	48.4	±8.1
B5	103.4	±36.0	39	±9.3

TABLE II
VARIOUS INITIAL STATES WITH LIMITED ACTION SET

The second set of five clockwork instances *(B1-B5)* were designed to be remarkably similar to the first five. The number of lights and the number of lights settings are still set to twelve and two respectively. The initial states for each instance are also the same as their *(A)* counterparts. What makes this set of instances different is the removal of $(3, 0)$ from the action set. The purpose of these instances is to demonstrate that the inclusion of inconsequential actions that don't alter the L-component of a state ultimately make for easier puzzle instances. The results for these instances are shown in Table II. Comparison between each numbered instance supports the suggestion that $(3, 0)$ simplified the puzzle for the EA. This result was also found by the panel of human players who found puzzles more frustrating when $(3, 0)$ was removed from the action set.

The third set of clockwork puzzle instances kept the number of lights at twelve. The action set $\{(7, \sigma^7 e_0), (5, \sigma^5 e_0)\}$

	Iter.	Iter. 95% CI	Length	Length 95% CI
C1	1.0	±0.6	24	±0
C2	186	±84.4	29.3	± 8.6
C3	778.5	±2315.3	41.7	±8.6
C4	3398.1	±1551.6	49.9	±10.9
C5	14536.5	±3454.4	27.3	± 18.1
C6	13899.9	±3839.9	15.3	±10.9

TABLE III
VARIOUS INITIAL STATES WITH THREE LIGHT SETTINGS

	Iter.	Iter. 95% CI	Length	Length 95% CI
D1	0.9	±0.6	30.3	±9.9
A5	47.1	±21.6	37.4	±10.5
D2	6138.4	±3450	46.8	± 13.5
D3	15801	±3038.3	22.3	±16.0

TABLE IV
"NOISY" INITIAL STATES WITH VARIABLE NO. OF LIGHTS

	Iter.	Iter. 95% CI	Length	Length 95% CI
E1	46.2	±18.5	53	±9.8
E2	43.1	±16.8	52.8	±11.4
E3	22.1	±7.6	36.7	±11.1
E4	33.3	±18.3	49.6	±12.3
E5	12.1	±4.4	43.6	±13.3

TABLE V
"NOISY" INITIAL STATES WITH VARYING ACTION SETS BASED ON Q

	Iter.	Iter. 95% CI	Length	Length 95% CI
F1	4215.6	±3014.2	50.9	± 14.2
F2	5026.9	±3127.7	35.8	±11.9
F3	6705.25	±3888.6	39.6	±16.1
F4	1680.25	± 1941.9	49.35	± 15.8

TABLE VI
RESULTS FROM ACTION SETS WITH IRREGULAR MOVES

also remains. The newly introduced difference is that the set of states used by the instance is now $W_{12,3}$. The initial states have also been altered to reflect the more complex states that one may start with as the number of light settings increases.

- *(C1)* All lights begin at the setting furthest from that of the light setting which identifies the goal.
- *(C2)* Starting from the arrangement *(C1)*, the even-numbered lights have their setting incremented by one.
- *(C3)* Starting from the arrangement *(C2)*, the numbered lights that can be divided evenly by three have their setting incremented by one.
- *(C4)* Starting from arrangement *(C1)*, a single light at the seventh position was set to the goal position.
- *(C5)* "Noisy" mix of light settings.
- *(C6)* "Noisy" mix of light settings, but with five light settings instead of three.

By referring to Table III, one can see that the EA took much longer to solve this set of instances than the first two sets of instances; this was also the first set of instances where failure was recorded as an outcome for a run. The human panel found these instances very difficult compared to the first two sets; almost every member abandoned their attempts for instances *(C4-C6)*.

In a similar fashion of expansion, the fourth set of instances altered the number of lights on the clock while fixing the number of light settings at two and using the same action set as the second and third instances; the initial states for these instances were all designed to be "noisy". For each instance *(D1,D2,D3)* in this set, the number of lights were six, eighteen, and twenty-four respectively; instance *(A5)* can also be a member of this set due to its similar arrangement. The results gathered in Table IV gave the predictable result that more lights for an instance equated to the EA requiring more time to create a solution. The panel of players also found that increasing the number of lights made for a more difficult puzzle; however, instances with $c > 2$ proved to be far more frustrating.

This leads to the fifth set of instances. These instances used the states $W_{12,2}$. The initial state for each instance was some noisy arrangement of lights settings. Each instance altered the value of Q computed from their action sets as discussed in Section IV; these action sets contained two moves: both with an L-component of $\sigma^k e_0$. What differed was the two distinct k-values of the actions.

- *(E1)* k-components of two and one for the actions; Q value of one. Instance contains 49152 reachable states.
- *(E2)* k-components of three and one for the actions; Q value of two. Instance contains 24576 reachable states.
- *(E3)* k-components of four and one for the actions; Q value of three. Instance contains 12288 reachable states.
- *(E4)* k-components of five and one for the actions; Q value of four. Instance contains 6144 reachable states.
- *(E5)* k-components of seven and one for the actions; Q value of six. Instance contains 3072 reachable states.

The results from Table V appear to suggest that as the number of reachable states decreases: the easier the instance is for EA; however, the differences between these instances are close to negligible compared to earlier results. The panel of human players reached a similar conclusion as they barely noted any difference in puzzle difficulty between the five instances.

The last set of problem instances attempted to consider how puzzle difficulty may be affected when one includes actions in their set different from those used in the previous five instances. These instances all used the same initial state, number of lights, and number of light settings as that of *(C3)*; their action sets were also like that of instance *(C3)*, but each separate instance made an addition.

- *(F1)* Added the action $(3, \sigma^3(2e_0))$.
- *(F2)* Added the action $(3, \sigma^3(e_2 + e_{11}))$.
- *(F3)* For this instance, both of the actions in *(F1)* and *(F2)* are added to the set.
- *(F4)* Two identical actions which both did nothing were added to the action set.

Table VI shows the results of these irregular action sets; the observations appear to suggest that the EA gets easily confused when presented with an abundance of unusual choices.

The last instance *(F4)* was included in order to showcase this particular failing of the EA. The two added moves did

nothing but occupy space in the ring and the EA wound up taking more iterations on average than it did for the similar instance *(C3)* to find a solution. The human players found these instances easier as they took advantage of irregular moves when possible and disregarded them otherwise.

VI. Conclusions

From the results gathered by Section V, there are some general results about clockwork puzzle instance difficulty which can be summarized. It appeared that each of the parameters that contributed towards a clockwork puzzle instance contribute to the measured difficulty of an instance.

The most plainly obvious result is demonstrated by the third, fourth, and fifth set of instances: as one increases the number of reachable states that exist within a clockwork puzzle instance, the harder that the puzzle instance is for both humans and the EA. This increase in the number of states can either be done by increasing n, increasing c, or by carefully selecting an action set and controlling the value of Q; the last of these options seemed to produces no difference to humans, but it is speculated that further investigation with Q on larger values of c may yield more pronounced differences in perceived difficulty. It makes intuitive sense that instances with a larger absolute number of reachable states would be more difficult to a human player as navigating through a large space of clock states would prove to be challenging; the player would on average require a longer string of actions to create a solution which introduces more chances for making an error on their part.

This leads to the conclusion about the impact initial states have on instance difficulty as shown by the first three sets. It does not come as much surprise that initial states which are highly patterned can be more easily solved than instances whose initial states are comprised of a noisy mix of light settings; the EA which favors generating highly patterned strings has a more difficult time solving these instances and needs to gradually develop a very specific string in order to solve them properly. The actual length of solution required to solve the puzzle can be deemed inconsequential so long as the actual actions which make up the string don't seem obvious to the player at their current state in the puzzle; manipulating the initial state and the goal states accordingly can confuse a player as they attempt to make a solution. This notion links back to Kotovosky's results on Tower of Hanoi which showed other discrepancies between human and computer reasoning on a puzzle

Putting aside initial states: the action set of an instance also has an impact on difficulty. The first two sets of instances in Section V show that the inclusion of actions which allow one to move the clock hand while not affecting any light settings creates an easier puzzle instance. The panel of human players felt that the inclusion of this action almost acted as "training wheels" for the puzzle as they were afforded far more leniency in their solution.

On the other side of difficulties relating to the action set: it appears that the inclusion of additional unusual actions to the set create an instances which are more difficult. This result was not shared by the panel of human players; when they have an unusual move at their disposal, the player only used it when it would be a clear benefit to them and avoided it otherwise.

Briefly looking at the length of solutions made by DWP, it appears that a wide variety of solution lengths were obtained by the algorithm. This neat feature showcases the myriad of ways one may attempt to approach the puzzle. The panel of human players had similar results as each presented slightly different solutions to the same instance. Difficulty could be added to a puzzle by enforcing a limit on the number of moves made by the player.

Future work will hope to look at even more clockwork puzzle instances: as both mathematical objects and as puzzles. In the case of the former, there is still much work to be done in succinctly proving which clockwork puzzle instances are solvable. At this time, it is still not known whether an instance is solvable when its action set is solely comprised of unusual actions that don't have L-components of either $\sigma^k e_0$ or all zeroes. In practice, a separate algorithm could just generate all possible states that are obtainable when starting from each goal state and using the action set of an instance; an initial state could then be picked from this set of states and said instance would be guaranteed as solvable. This approach may be adopted for future use, but whether there exists a sound mathematical proof behind what properties permit an instance to be solvable is still of acute interest. It is hoped that as more results are uncovered for the clockwork puzzle, they may hold applications in other areas of math as well.

For now, the novel clockwork puzzle is simply meant to be used as a fun puzzle that could be included as a possible problem in a testing suite for general algorithms which are being applied towards combinatorial problems. The puzzle would also be suitable as a feature within video games in order to test a player's spatial reasoning ability.

References

[1] George Marx, Eva Gajzago, and Peter Gnadig. The universe of rubik's cube. *European Journal of Physics*, 3(1):39, 1982.

[2] Peter Buneman et al. The towers of hanoi problem. *Information Processing Letters*, 10(4,5), 1980.

[3] Harry Kesten. On the number of self-avoiding walks. *Journal of Mathematical Physics*, 4(7):960–969, 1963.

[4] Daniel Ashlock and Christoph Salge. Automatic generation of level maps with the do what's possible representation. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[5] Justin Shonfeld and Daniel Ashlock. Flow of control in linear genetic programming. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1175–1182. IEEE, 2015.

[6] Daniel Ashlock, Sierra Gillis, Andrew McEachern, and Jeffrey Tsang. The do what's possible representation. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1586–1593. IEEE, 2016.

[7] William M Spears. Adapting crossover in evolutionary algorithms. In *Evolutionary programming*, pages 367–384, 1995.

[8] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[9] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[10] Kenneth Kotovsky, John R Hayes, and Herbert A Simon. Why are some problems hard? evidence from tower of hanoi. *Cognitive psychology*, 17(2):248–294, 1985.