**Introduction**

The [*Forgotten Age* cycle](https://www.fantasyflightgames.com/en/products/arkham-horror-the-card-game/products/forgotten-age/) of Arkham Horror is at a close and Fantasy Flight Games already [announced the next cycle, *The Circle Undone*](https://www.fantasyflightgames.com/en/products/arkham-horror-the-card-game/products/circle-undone/). Not only that, they’ve announced [two](https://www.fantasyflightgames.com/en/products/arkham-horror-the-card-game/products/secret-name/) [mythos](https://www.fantasyflightgames.com/en/products/arkham-horror-the-card-game/products/wages-sin/) packs at a rate that… surprised me. A new cycle announcement and two mythos pack announcements in less than two months? Am I the only one who finds the new pace of announcements surprising? Perhaps that means they want to get product out at a faster pace?

Eh, enough speculation. I wrote about Arkham Horror before, [analyzing Olive McBride](https://ntguardian.wordpress.com/2018/08/13/evaluating-olive-mcbride-arkham-horror-lcg-chaos-bag-simulator-r/) specifically. This analysis (despite errors in the initial publication) was well received, even earning me a shoutout from my favorite Arkham-related YouTube channel.

In the announcement of the mythos pack [*The Wages of Sin*](https://www.fantasyflightgames.com/en/news/2018/11/30/the-wages-of-sin/), another mathematically interesting card was spoiled: [Henry Wan](https://images-cdn.fantasyflightgames.com/filer_public/b6/86/b68693c8-c1f8-497c-be63-d4aacb3768c2/ahc31_card_henry-wan.png), seen below.



**Henry Wan’s Challenge**

Designing new allies for Arkham Horror is very hard because there can effectively only be one ally in a deck and there are many good allies already released, many of them in the [core set](https://www.fantasyflightgames.com/en/products/arkham-horror-the-card-game/products/arkham-horror-card-game/). Henry Wan, specifically, is competing with [Leo de Luca](https://arkhamdb.com/card/01048), who competes with [Dr. Milan Christopher](https://arkhamdb.com/card/01033) for the title of “Best Ally”. Cards like [Charisma](https://arkhamdb.com/find?q=charisma) help the problem, but only if you plan on running multiple allies and are willing to pay the experience points for it.



Can Henry Wan compete with Leo de Luca? That strongly depends on how good his ability is. Actions are a precious commodity in Arkham Horror; this is why Leo de Luca is considered such a great card. Card draw and resource gain *can* help action economy, especially in a spendthrift class such as the Rogue (green) class, but it often takes many resources to compensate for a lost action.



Consider, for instance, [Father Mateo’s](https://arkhamdb.com/find?q=father+mateo) Elder Sign ability; gain an extra action, or a card and a resource. As a point of reference, players can draw a card *or* gain a resource for one of their actions, so a raw evaluation would say that drawing a card *and* gaining a resource is actually worth two actions and thus is better than just getting a free action. But I feel most of the time people use Father Mateo’s elder sign effect to gain the additional action rather than the card and resource (though choosing the latter effect is far from rare). In fact, I think that a single action could be valued at *three* resources, based only on the fact that when a player draws [Emergency Cache](https://arkhamdb.com/card/01088) they will eagerly play it. When viewed from this perspective, Leo de Luca pays for himself after about three turns, and drawing him early gives an investigator a major boost in a scenario.



**Gambling with Henry Wan**

Henry Wan will thus live or die based on how strong his ability is. That said, “strong” depends on how well a player can use his ability, which is not a trivial task.

Make no mistake: Henry Wan is a gambler’s card (which fits the Rogue theme very well). Not only does a player gamble the resources spent on him, they gamble the action spent to trigger his ability; heck, using a deck slot on him is a gamble! A player thus will gain value from him *only* if they use him optimally.

Optimal play is not trivially determined, but fortunately Henry Wan’s ability is easy to model mathematically if you’re familiar with [Markov chains](https://en.wikipedia.org/wiki/Markov_chain). Wait, are most people *not* familiar with Markov chains? Oh, I didn’t know that. Oh well, maybe they’ll learn something from what follows. I’ll do my best to make it simple.

From here on, I consider drawing a card or gaining a resource with Henry Wan as equivalent; I’ll simply imagine that we’re trying to gain resources using his ability. Henry Wan’s ability calls on players to institute a policy for playing him of the following form:

**After X draws, take your winnings; do not draw anymore.**

Our job is thus to choose X so that we maximize the *expected* resource gain (in the [probabilistic sense of expectation](https://en.wikipedia.org/wiki/Expected_value).)

I’m going to call utilizing Henry Wan’s ability a single “game”. Here’s how I view the game: the chaos bag is filled with tokens labeled either “S” or “F”, with every “F” being one of the icon tokens mentioned in Henry Wan’s ability. When an “S” is drawn, the game continues, while the game ends the moment an “F” is drawn. Every time we draw an additional “S”, there is one fewer “S” in the bag, and the odds of drawing an “F” increase; that said, our total winnings increase with each “S” we draw.

The game ends when either an “F” is drawn or the policy is triggered. Our winnings depend on which of these outcomes we find ourselves. If it’s the former, our winnings are 0, while if the latter, our winnings are X. Thus it’s easy to see (if you’re familiar with probability) that the expected winnings for any given policy is X times the probability of winning with the chosen policy: X p_X, if you prefer (with p_Xbe the probability of not failing using the policy of ending after X draws). We thus want to pick X that maximizes X p_X.

Calculating p_Xcalls for the Markov chain. Below is the chain I imagine:

1. The initial state is state 0, representing zero draws. There are also states numbered 1 to X, and a state F.
2. If the chain is at state i < X, the chain moves to state i + 1with probability p_ior to state F with probability 1 - p_i.
3. Both state X and state F are [absorbing states](https://en.wikipedia.org/wiki/Absorbing_Markov_chain). (Once entered, the chain does not leave the state; in other words, the "game" ends.)

The problem now is to calculate the probability the chain is absorbed into state X. The solution of ending in a particular absorbing state is well known (and given in the above link to Wikipedia).

No special trick for finding a maximizing X is necessary once we know how to solve this problem for any X; just list out all possible policies (there's only finitely many we need to worry about, and the number doesn't exceed 20 most of the time) and the expected winnings and pick the X maximizing this number.

The maximizing policy depends on what's in the chaos bag. Shocking, right? That said, this is an important point; each campaign/scenario/difficulty level has its own chaos bag, and thanks to cards with the [**seal**](https://arkhamdb.com/find?q=x%3Aseal) keyword, the chaos bag can be changed *during* a scenario, perhaps to either the benefit or detriment of Henry Wan. Fortunately, the "S" and "F" language makes modelling the contents of the chaos bag so simple, we can create two-dimensional tables depending only on the number of "S's" and "F's" in the bag and those tables will cover nearly every scenario an investigator will encounter.

The script below (which can be made executable on Unix systems with R installed) can be used for generating such tables.

#!/usr/bin/Rscript

################################################################################

# ArkhamHorrorHenryWanTableGenerator.R

################################################################################

# 2018-12-02

# Curtis Miller

################################################################################

# This is a one-line description of the file.

################################################################################

# optparse: A package for handling command line arguments

if (!suppressPackageStartupMessages(require("optparse"))) {

install.packages("optparse")

require("optparse")

}

################################################################################

# FUNCTIONS

################################################################################

#' Henry Wan Policy Calculator

#'

#' Calculates important quantities for optimal play with Henry Wan

#'

#' @param s The number of "S" (or "success") tokens in the bag

#' @param f The number of "F" (or "failure") tokens in the bag

#' @param olive If \code{TRUE}, the first draw is done with Olive McBride

#' @param out If \code{"X"}, return the optimal stopping time (default); if

#' \code{"EV"}, return the expected winnings of the optimal policy;

#' if \code{"P"}, return the probability of success of the optimal

#' policy

#' @return Numeric depending on the value of the parameter \code{out}

#' @examples

#' wan\_policy\_calculator(11, 5)

wan\_policy\_calculator <- function(s, f, olive = FALSE,

out = c("X", "EV", "P")) {

out <- out[[1]]

policies <- (ifelse(olive, 2, 1)):s # Candidate X values

policy\_probs <- sapply(policies, function(X) {

# Set up transition matrix of Markov chain

P <- 0 \* diag(X + 2)

rownames(P) <- c(0:X, "F")

colnames(P) <- rownames(P)

P[c(X, "F"), c(X, "F")] <- diag(2)

transient\_states <- ifelse(X > 1, list(c("0", 1:(X - 1))), "0")[[1]]

P[transient\_states, "F"] <- f/(s + f - (0:(X - 1)))

if (olive) {

if (s + f < 3 | X == 1) {

stop("X or chaos bag doesn't make sense with Olive!")

}

# Failure with Olive is modeled with a hypergeometric RV, with drawing one

# or fewer "S's"

P["0", "F"] <- phyper(1, m = s, n = f, k = 3)

# The state 1 is effectively removed when Olive is used

transient\_states <- transient\_states[-2]

P <- P[-2, -2]

# TODO: curtis: OLIVE IMPELENTATION -- Sun 02 Dec 2018 11:05:17 PM MST

}

if (X > 1) {

if (olive & X == 2) {

P["0", "2"] <- 1 - P["0", "F"]

} else {

P[transient\_states, as.character((ifelse(olive, 2, 1)):X)] <- diag(

c(1 - P[transient\_states, "F"]))

}

} else {

P["0", "1"] <- 1 - P["0", "F"]

}

# Compute absorption probability

R <- P[transient\_states, c(X, "F")]

Q <- P[transient\_states, transient\_states, drop = FALSE]

N <- solve(diag(nrow(Q)) - Q)

B <- N %\*% R

B[1,1][[1]]

})

X <- which.max(policy\_probs \* policies)

if (out == "X") {

policies[[X]]

} else if (out == "EV") {

policies[[X]] \* policy\_probs[[X]]

} else if (out == "P") {

policy\_probs[[X]]

} else {

stop(paste("Don't know how to handle out =", out))

}

}

wan\_policy\_calculator <- Vectorize(wan\_policy\_calculator, c("s", "f"))

################################################################################

# MAIN FUNCTION DEFINITION

################################################################################

main <- function(olive = FALSE, value = FALSE, prob = FALSE, digits = 2,

lower\_s = 5, upper\_s = 20, lower\_f = 0, upper\_f = 8,

help = FALSE) {

# This function will be executed when the script is called from the command

# line; the help parameter does nothing, but is needed for do.call() to work

library(pander)

sl <- lower\_s

su <- upper\_s

fl <- lower\_f

fu <- upper\_f

out <- "X"

if (value) {out <- "EV"}

if (prob) {out <- "P"}

wan\_table <- outer(sl:su, fl:fu, FUN = function(r, c) {

wan\_policy\_calculator(r, c, olive = olive, out = out)

})

rownames(wan\_table) <- sl:su

colnames(wan\_table) <- fl:fu

wan\_table <- round(wan\_table, digits = digits)

pandoc.table(wan\_table, style = "rmarkdown")

}

################################################################################

# INTERFACE SETUP

################################################################################

if (sys.nframe() == 0) {

cl\_args <- parse\_args(OptionParser(

description = paste("Generates tables describing optimal policies",

"for playing with the card Henry Wan in",

"Arkham Horror: The Card Game (number of icon",

"tokens in bag are columns; non-icon rows)."),

option\_list = list(

make\_option(c("--olive", "-o"), action = "store\_true",

default = FALSE,

help = "The first draw is done with Olive"),

make\_option(c("--value", "-v"), action = "store\_true",

default = FALSE,

help = paste("Report expected value rather than",

"optimal stopping policy")),

make\_option(c("--prob", "-p"), action = "store\_true",

default = FALSE,

help = paste("Report success probability of optimal",

"stopping policy rather than the",

"optimal stopping policy itself")),

make\_option(c("--digits", "-d"), type = "integer", default = 2,

help = "Number of digits for rounding"),

make\_option(c("--lower-s", "-s"), type = "integer", default = 5,

help = "Lowest considered number of non-icon tokens"),

make\_option(c("--upper-s", "-w"), type = "integer", default = 20,

help = "Highest considered number of non-icon tokens"),

make\_option(c("--lower-f", "-f"), type = "integer", default = 0,

help = "Lowest considered number of icon tokens"),

make\_option(c("--upper-f", "-r"), type = "integer", default = 8,

help = "Highest number of icon tokens")

)))

cl\_args <- cl\_args[c("olive", "value", "prob", "digits", "lower-s", "upper-s",

"lower-f", "upper-f", "help")]

names(cl\_args) <- c("olive", "value", "prob", "digits", "lower\_s", "upper\_s",

"lower\_f", "upper\_f", "help")

do.call(main, cl\_args)

}

With the above script I can make the following three tables. The columns represent the number of (bad) icon tokens in the bag, while rows represent the number of other tokens in the bag. The first table is the optimal stopping policy; the second, the probability of success of the optimal stopping policy; and the third, the expected winnings of the optimal policy (which is the product of the previous two tables).

**Optimal stopping policy**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 5 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| **6** | 6 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| **7** | 7 | 4 | 3 | 2 | 2 | 2 | 1 | 1 | 1 |
| **8** | 8 | 4 | 3 | 3 | 2 | 2 | 2 | 1 | 1 |
| **9** | 9 | 5 | 3 | 3 | 2 | 2 | 2 | 2 | 1 |
| **10** | 10 | 5 | 4 | 3 | 3 | 2 | 2 | 2 | 2 |
| **11** | 11 | 6 | 4 | 3 | 3 | 2 | 2 | 2 | 2 |
| **12** | 12 | 6 | 4 | 4 | 3 | 3 | 2 | 2 | 2 |
| **13** | 13 | 7 | 5 | 4 | 3 | 3 | 2 | 2 | 2 |
| **14** | 14 | 7 | 5 | 4 | 3 | 3 | 3 | 2 | 2 |
| **15** | 15 | 8 | 6 | 4 | 3 | 3 | 3 | 2 | 2 |
| **16** | 16 | 8 | 6 | 5 | 4 | 3 | 3 | 3 | 2 |
| **17** | 17 | 9 | 6 | 5 | 4 | 3 | 3 | 3 | 2 |
| **18** | 18 | 10 | 6 | 5 | 4 | 3 | 3 | 3 | 2 |
| **19** | 19 | 10 | 7 | 5 | 4 | 4 | 3 | 3 | 3 |
| **20** | 20 | 11 | 7 | 5 | 5 | 4 | 3 | 3 | 3 |

**Probability of success of optimal policy**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 1 | 0.5 | 0.48 | 0.36 | 0.56 | 0.5 | 0.45 | 0.42 | 0.38 |
| **6** | 1 | 0.57 | 0.54 | 0.42 | 0.33 | 0.55 | 0.5 | 0.46 | 0.43 |
| **7** | 1 | 0.5 | 0.42 | 0.47 | 0.38 | 0.32 | 0.54 | 0.5 | 0.47 |
| **8** | 1 | 0.56 | 0.47 | 0.34 | 0.42 | 0.36 | 0.31 | 0.53 | 0.5 |
| **9** | 1 | 0.5 | 0.51 | 0.38 | 0.46 | 0.4 | 0.34 | 0.3 | 0.53 |
| **10** | 1 | 0.55 | 0.42 | 0.42 | 0.33 | 0.43 | 0.38 | 0.33 | 0.29 |
| **11** | 1 | 0.5 | 0.46 | 0.45 | 0.36 | 0.46 | 0.4 | 0.36 | 0.32 |
| **12** | 1 | 0.54 | 0.49 | 0.36 | 0.39 | 0.32 | 0.43 | 0.39 | 0.35 |
| **13** | 1 | 0.5 | 0.43 | 0.39 | 0.42 | 0.35 | 0.46 | 0.41 | 0.37 |
| **14** | 1 | 0.53 | 0.46 | 0.42 | 0.45 | 0.38 | 0.32 | 0.43 | 0.39 |
| **15** | 1 | 0.5 | 0.4 | 0.45 | 0.47 | 0.4 | 0.34 | 0.45 | 0.42 |
| **16** | 1 | 0.53 | 0.43 | 0.38 | 0.38 | 0.42 | 0.36 | 0.32 | 0.43 |
| **17** | 1 | 0.5 | 0.46 | 0.4 | 0.4 | 0.44 | 0.38 | 0.34 | 0.45 |
| **18** | 1 | 0.47 | 0.48 | 0.42 | 0.42 | 0.46 | 0.4 | 0.35 | 0.47 |
| **19** | 1 | 0.5 | 0.43 | 0.44 | 0.44 | 0.36 | 0.42 | 0.37 | 0.33 |
| **20** | 1 | 0.48 | 0.45 | 0.46 | 0.36 | 0.38 | 0.44 | 0.39 | 0.35 |

**Expected winnings of optimal policy**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 5 | 1.5 | 0.95 | 0.71 | 0.56 | 0.5 | 0.45 | 0.42 | 0.38 |
| **6** | 6 | 1.71 | 1.07 | 0.83 | 0.67 | 0.55 | 0.5 | 0.46 | 0.43 |
| **7** | 7 | 2 | 1.25 | 0.93 | 0.76 | 0.64 | 0.54 | 0.5 | 0.47 |
| **8** | 8 | 2.22 | 1.4 | 1.02 | 0.85 | 0.72 | 0.62 | 0.53 | 0.5 |
| **9** | 9 | 2.5 | 1.53 | 1.15 | 0.92 | 0.79 | 0.69 | 0.6 | 0.53 |
| **10** | 10 | 2.73 | 1.7 | 1.26 | 0.99 | 0.86 | 0.75 | 0.66 | 0.59 |
| **11** | 11 | 3 | 1.85 | 1.36 | 1.09 | 0.92 | 0.81 | 0.72 | 0.64 |
| **12** | 12 | 3.23 | 1.98 | 1.45 | 1.18 | 0.97 | 0.86 | 0.77 | 0.69 |
| **13** | 13 | 3.5 | 2.14 | 1.57 | 1.26 | 1.05 | 0.91 | 0.82 | 0.74 |
| **14** | 14 | 3.73 | 2.29 | 1.68 | 1.34 | 1.13 | 0.96 | 0.87 | 0.79 |
| **15** | 15 | 4 | 2.43 | 1.78 | 1.41 | 1.2 | 1.03 | 0.91 | 0.83 |
| **16** | 16 | 4.24 | 2.59 | 1.88 | 1.5 | 1.26 | 1.09 | 0.95 | 0.87 |
| **17** | 17 | 4.5 | 2.74 | 2 | 1.59 | 1.32 | 1.15 | 1.01 | 0.91 |
| **18** | 18 | 4.74 | 2.87 | 2.11 | 1.67 | 1.38 | 1.21 | 1.06 | 0.94 |
| **19** | 19 | 5 | 3.03 | 2.21 | 1.75 | 1.46 | 1.26 | 1.12 | 0.99 |
| **20** | 20 | 5.24 | 3.18 | 2.3 | 1.82 | 1.53 | 1.32 | 1.17 | 1.04 |

I view column 6, row 11 as the "typical" scenario, and the conclusion is this: *you'd be better off just grabbing a resource/drawing a card the usual way than by using Henry Wan!* Not only is Henry Wan worse than Leo de Luca, *he's worse than gaining resources with a regular action!*

Granted, there are cards with the *seal* keyword that can help improve the odds. But one must ask whether the opportunity cost of playing those cards is worth it. Perhaps the benefits of a favorable chaos bag for skill tests plus better Henry Wan games would give the investigators a *teeny tiny* edge… after a hell of a lot of work and lucky draw. That said, I'm sure there's much easier ways to play the game that are also more fun.

**Henry Wan and Olive McBride: A Match Made in Heaven?**



When Henry Wan was announced, people considered pairing him up with Olive McBride, who's ability works "when you would reveal a chaos token". Any investigator that can take both Mystic (purple) and Rogue (green) cards (including Sefina Rousseau and all [Dunwich investigators](https://arkhamdb.com/find?q=y%3A2+t%3Ainvestigator&sort=faction&view=list&decks=all&spoilers=hide); I don't count [Lola Hayes](https://arkhamdb.com/card/03006) since, while she can include both cards in her deck, using them together may not be possible) can include these two cards in the same deck.

I'll always assume that Olive's ability is utilized on the first draw. When using Olive with Henry, one can get two tokens drawn without either of them being a bad icon that ends the "game". Thus Olive boosts the success rate and the ultimate payout.

Having Olive and Henry out at the same time is extremely difficult; first, you'd have to have charismas to accomodate them, then draw them both in a game at reasonable times. The likelihood of getting the combo out is low and comes with significant opportunity costs.

That said, when Olive is out, she provides Henry enough of a boost to make him playable. The following tables account for Olive's effect (see the code for how) on the first draw but otherwise match up with the earlier tables.

**Optimal stopping policy (with Olive McBride's effect)**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 5 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **6** | 6 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **7** | 7 | 4 | 3 | 2 | 2 | 2 | 2 | 2 | 2 |
| **8** | 8 | 4 | 3 | 2 | 2 | 2 | 2 | 2 | 2 |
| **9** | 9 | 5 | 3 | 3 | 2 | 2 | 2 | 2 | 2 |
| **10** | 10 | 6 | 4 | 3 | 3 | 2 | 2 | 2 | 2 |
| **11** | 11 | 6 | 4 | 3 | 3 | 2 | 2 | 2 | 2 |
| **12** | 12 | 7 | 4 | 3 | 3 | 3 | 2 | 2 | 2 |
| **13** | 13 | 7 | 5 | 4 | 3 | 3 | 2 | 2 | 2 |
| **14** | 14 | 7 | 5 | 4 | 3 | 3 | 3 | 2 | 2 |
| **15** | 15 | 8 | 6 | 4 | 3 | 3 | 3 | 2 | 2 |
| **16** | 16 | 8 | 6 | 5 | 4 | 3 | 3 | 3 | 2 |
| **17** | 17 | 9 | 6 | 5 | 4 | 3 | 3 | 3 | 2 |
| **18** | 18 | 10 | 6 | 5 | 4 | 3 | 3 | 3 | 3 |
| **19** | 19 | 10 | 7 | 5 | 4 | 4 | 3 | 3 | 3 |
| **20** | 20 | 11 | 7 | 5 | 4 | 4 | 3 | 3 | 3 |

**Probability of success of optimal policy (with Olive McBride's effect)**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 1 | 0.75 | 0.86 | 0.71 | 0.6 | 0.5 | 0.42 | 0.36 | 0.31 |
| **6** | 1 | 0.8 | 0.89 | 0.77 | 0.67 | 0.58 | 0.5 | 0.44 | 0.38 |
| **7** | 1 | 0.67 | 0.65 | 0.82 | 0.72 | 0.64 | 0.56 | 0.5 | 0.45 |
| **8** | 1 | 0.71 | 0.7 | 0.85 | 0.76 | 0.69 | 0.62 | 0.55 | 0.5 |
| **9** | 1 | 0.62 | 0.74 | 0.61 | 0.8 | 0.73 | 0.66 | 0.6 | 0.55 |
| **10** | 1 | 0.56 | 0.59 | 0.65 | 0.55 | 0.76 | 0.7 | 0.64 | 0.59 |
| **11** | 1 | 0.6 | 0.63 | 0.68 | 0.59 | 0.79 | 0.73 | 0.67 | 0.62 |
| **12** | 1 | 0.55 | 0.66 | 0.71 | 0.62 | 0.54 | 0.75 | 0.7 | 0.66 |
| **13** | 1 | 0.58 | 0.56 | 0.56 | 0.64 | 0.57 | 0.78 | 0.73 | 0.68 |
| **14** | 1 | 0.62 | 0.59 | 0.59 | 0.67 | 0.6 | 0.53 | 0.75 | 0.71 |
| **15** | 1 | 0.57 | 0.51 | 0.61 | 0.69 | 0.62 | 0.56 | 0.77 | 0.73 |
| **16** | 1 | 0.6 | 0.54 | 0.51 | 0.54 | 0.64 | 0.58 | 0.53 | 0.75 |
| **17** | 1 | 0.56 | 0.56 | 0.53 | 0.57 | 0.66 | 0.6 | 0.55 | 0.77 |
| **18** | 1 | 0.53 | 0.59 | 0.55 | 0.59 | 0.68 | 0.62 | 0.57 | 0.52 |
| **19** | 1 | 0.56 | 0.52 | 0.57 | 0.6 | 0.53 | 0.64 | 0.59 | 0.54 |
| **20** | 1 | 0.53 | 0.55 | 0.59 | 0.62 | 0.55 | 0.66 | 0.61 | 0.56 |

**Expected winnings of optimal policy (with Olive McBride's effect)**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | 5 | 2.25 | 1.71 | 1.43 | 1.19 | 1 | 0.85 | 0.73 | 0.63 |
| **6** | 6 | 2.4 | 1.79 | 1.55 | 1.33 | 1.15 | 1 | 0.87 | 0.77 |
| **7** | 7 | 2.67 | 1.96 | 1.63 | 1.44 | 1.27 | 1.13 | 1 | 0.89 |
| **8** | 8 | 2.86 | 2.1 | 1.7 | 1.53 | 1.37 | 1.23 | 1.11 | 1 |
| **9** | 9 | 3.12 | 2.21 | 1.83 | 1.59 | 1.45 | 1.32 | 1.2 | 1.09 |
| **10** | 10 | 3.33 | 2.38 | 1.95 | 1.65 | 1.52 | 1.39 | 1.28 | 1.18 |
| **11** | 11 | 3.6 | 2.52 | 2.04 | 1.76 | 1.57 | 1.46 | 1.35 | 1.25 |
| **12** | 12 | 3.82 | 2.64 | 2.12 | 1.85 | 1.62 | 1.51 | 1.41 | 1.31 |
| **13** | 13 | 4.08 | 2.8 | 2.24 | 1.93 | 1.71 | 1.56 | 1.46 | 1.37 |
| **14** | 14 | 4.31 | 2.95 | 2.36 | 2.01 | 1.79 | 1.6 | 1.51 | 1.42 |
| **15** | 15 | 4.57 | 3.07 | 2.45 | 2.07 | 1.86 | 1.67 | 1.55 | 1.46 |
| **16** | 16 | 4.8 | 3.24 | 2.54 | 2.17 | 1.93 | 1.75 | 1.58 | 1.5 |
| **17** | 17 | 5.06 | 3.38 | 2.66 | 2.26 | 1.99 | 1.81 | 1.65 | 1.54 |
| **18** | 18 | 5.29 | 3.51 | 2.77 | 2.34 | 2.04 | 1.87 | 1.71 | 1.57 |
| **19** | 19 | 5.56 | 3.67 | 2.87 | 2.42 | 2.12 | 1.92 | 1.77 | 1.63 |
| **20** | 20 | 5.79 | 3.82 | 2.96 | 2.49 | 2.2 | 1.97 | 1.82 | 1.69 |

Notice that when Olive is being used it's optimal to use Olive to get two tokens out (that you can pick) then end the "game". It seems that Olive does make Henry's ability profitable… albeit mildly. If we're using the heuristic that a card needs to pay for its own resource cost plus three for each action involved, I'd say that the combo would need at least eight turns to be profitable in a typical game… which is terrible.

Henry Wan is an expensive way to attempt to milk a little more value from Olive. Even with Olive I don’t think he’s worth the trouble.

**Conclusion: Gambling is Bad, Kids**

It's equally true in Arkham as it is in real life: gambling is better for the house than the gambler (with the house being the forces of the mythos, in this case). If you're looking to have fun gambling, Henry Wan is your card. If you're looking to win… look elsewhere.