

Algorithms for k/n Power-Hours

Ben Blum Dr. William Lovas, Ph.D.
Chris Martens Dr. Tom Murphy VII, Ph.D.

1 April 2012

Abstract

Drinking games, while a fun activities, can lead to memory leaks if performed to excess. In particular the Power Hour, in which a shot of beer is drunk every minute for an hour, may be modified to allow potentially arbitrarily customizably safely enjoyable consumption. We sketch some known solutions and avenues for future research. ALSO WE ARE DRINKING RIGHT NOW AND THIS PAPER WAS COMPLETED IN ONE HOUR

Keywords: alcohol in computer science, algorithms for the real world, chemically-assisted reasoning, drinking game theory, hyper-driven devices

1 Introduction

A *power hour*¹ is a drinking game in which participants drink a shot of beer every minute for an hour, usually based on musical cues.

Using the standard of one shot = one fluid ounce of 4% alcohol, a power hour equals approximately five beers total. For an average-mass human with a well-developed alcohol tolerance, the game results in a pleasant level of inebriation.²

¹Copyright © 2012 the Regents of the Wikiplia Foundation. Appears in SIGBOVIK 2012 with the permission of the Association for Computational Heresy; *IEEEEE!* press, Verlag-Verlag volume no. 0x40-2A. £0.00

¹Not to be confused with power sets, Powerade, Powerpuff Girls, or Mighty Morphin' Power Rangers

²Anonymous personal correspondance

However, in some cases, the game may result in uncomfortable levels of inebriation. Participants therefore may wish to reduce the total amount of alcohol consumed while still experiencing the process of collaborative inebriation.

If, say, a participant wants to drink half as much as everyone else, what options do they have available? One possibility is to simply drink once every other song. This is problematic for two reasons. The soft reason is that in the spirit of active participation, they would ideally like to take some action progressing their drunkenness at every song change. See 3 for a formal discussion of this condition. A more compelling reason is that humans in a state of ever-increasing inebriation probably cannot remember whether or not they drank last time due to impaired reasoning abilities³ Therefore, they must, as a finite state automoton, determine their course of action based solely on current state.

A known solution for the *half* power hour is to take a different action based on shot glass state: *fill* when the glass is empty and *drink* if the glass is full. This elegant solution achieves the goal of consuming half as much alcohol by the end but requires the participant only to observe the most recent state, then change that state in a single action.

The aim of this work is to generalize the 1/2 power hour to general k/n (with m participants).

We provide some preliminary results, but primarily we pose a call to action suggesting various avenues of research.

³A non-judgmental reconstruction of drunken logic. Robert J. Simmons. SIGBOVIK 2007. April 2007.

2 Desiderata

3 Desiderata

There are several properties we would like an algorithm to satisfy in order to be considered a proper power hour algorithm. In this section, we enumerate these desiderata along with illustrative examples that violate—and thus motivate—them. Without constraints, the space of potential power hour algorithms is too large to be meaningfully analyzed and understood; these desiderata serve to limit the space of possible algorithms to those that are sufficiently simple and adequate to be implemented in a real-world context.

In what follows, a power hour *player* is tasked with taking an *action* each *turn*. Typically, a *turn* occurs every minute. Each *action* involves some observation of the current state and some change of state. The classic power hour algorithm is for each action to be: *observe* the empty shot glass in front of you, *fill* that shot glass with beer, and *drink* that shot glass, re-establishing the state invariant for the next round.


3.1 Discretion

The first and most important desideratum is that of *discretion*: each player should drink only *one* shot per turn. Anyone who has participated in a power hour realizes the difficulty of drinking even a single shot late in the game, and we wish to exclude algorithms that require a player to exceed these natural human limits.

An obvious violation is the 2-power hour (= 120/60-power hour), where a player must pour and drink two shots each turn.

3.2 Simplicity

Relatedly, a power hour algorithm should consist of only *simple* actions. There is some leeway in defining precisely what it means for an action to be “simple”, but the purpose of this constraint is to ensure maximal physical safety and minimal broken glass (desires which are in synergistic harmony), so for example, inverting a full cup is disallowed, since the spilled beer

causes a sense of alarm, and stacking a cup on an up-cup is unpermitted (e.g., ) since an up-cup does not provide sufficient foundation for safe stacking.

3.3 Locality

For practical purposes, a player participating in a proper power hour can perform only very simple *observations*: to that end, we posit the desideratum of *locality*: a player’s action may depend only upon observing the state of the shot glass directly in front of them, and not, say, the state of some other player’s glass or some written memory. This desideratum represents a kind of “memory safety”: players need not have too much memory from turn to turn, since in our experience, they won’t.

3.4 Singularity

Simplicity and locality together suggest the desideratum of *singularity*: a player should have at most one shot glass in front of them at any given moment. Generalizations are possible: for instance, every shot glass in front of a player must have the same state (e.g., all filled, all empty, all inverted, etc.), but the resultant protocols become prohibitively complex due to the explosion of possible states and the necessity of maintaining state invariants on each action.

3.5 Liveness

Another desideratum is the property of *liveness*: we would like every player to drink infinitely often, in the limit. The goal here is to ensure that every player continues to enjoy in the camaraderie at all times, and to a certain extent to maintain their buzz at a smooth and constant level. Liveness does, however, rule out many potential interesting algorithms like the 0/60-power hour, the 1/60-power hour, etc.

3.6 Extensibility

In addition to bounding the lower limit of a player’s drinking (liveness, above), we would like to bound their upper limit: the desideratum of *extensibility* captures this idea. Extensibility dictates that for any

generalized extension of the hour to $n' > n$ minutes, there must be some further extension to $n'' \geq n'$ minutes such that after n'' minutes, a player has consumed k'' drinks such that $k''/n'' = k/n$, exactly. In other words, in the limit, a k/n -player must always have drunk k/n times, or be on the way to drinking k/n times. Extensibility rules out algorithms like the $58/60$ -power hour, the $59/60$ -power hour, and the $61/60$ -power hour, where the player has some initial sequence of actions before they enter their main loop.

In the case of non-trivial (*i.e.*, non-zero) power hours, extensibility is a strictly stronger constraint than liveness, since for any $0 < x \leq 1$, in the limit, a player will eventually have to drink to maintain a fraction of x drinks per turn.

3.7 Asynchrony

In the case of distributed power hour algorithms (as in Section 5.1 and 4), we posit the desideratum of *asynchrony*: a player's action should not depend on any coordination with other players. Formally, and practically speaking, asynchrony requires that a player's *action* during a turn depend only upon the *observations* that player could have made at the beginning of the turn. Otherwise, you know, things just get too, uh... complicated.

4 Known Results

Solo arrangements. In the solo case, we know exactly what power hours are possible and which are not. Let us exhaust these before moving onto the more difficult distributed case.

- $0/60$ Trivial, with multiple solutions. Start with \cup , never fill it, and never drink.
- $1/60$ Many solutions. For example, start with \wp . Drink on \wp , leave \cup upon seeing \cup .
- $2/60$ Start with \wp . On \wp , drink and leave \cup . On \cup , fill and drink, and leave \cap . On \cap , do nothing.
- $3/60$ Impossible. Would require four distinct states, but there are only three.
- $4\dots19/60$ Even more impossible. XXX is 19 actually impossible?
- $20/60$ Drinking one shot out of three. Start with \wp . On \wp , drink and leave \cup . On \cup , flip to \cap . On \cap , flip, fill, and leave \wp .
- $21\dots29/60$ Even more impossible. XX are 21, 29 actually impossible?
- $30/60$ Drinking every other shot. Start with \wp . On \wp , drink and leave \cup . On \cup , fill and leave \wp .
- $31\dots39/60$ Super impossible.
- $40/60$ Drinking two shots out of three. Start with \wp . On \wp , drink and leave \cup . On \cup , fill, drink, and flip to \cap . On \cap , flip, fill, and leave \wp .
- $41\dots57/60$ Totally impossible!
- $58/60$ Symmetric to the $2/60$ case. Start with \cap . On \cap , flip and leave \cup . On \cup , fill and leave \wp . On \wp , drink, fill, and leave \wp .
- $59/60$ Symmetric to the $1/60$ case. Start with \cup . On \cup , fill and leave \wp . On \wp , drink, fill, and leave \wp .
- $60/60$ Easy; this is a normal power-hour. Start with \cup . On \cup , fill, drink, and leave \cup .

Distributed algorithms. Generalizing to the distributed case unlocks many more possibilities. Even for just a small number of participants, it becomes quite difficult to exhaustively explore the possibilities. These algorithms are a class of finite state machines, probably excluding analytical approaches (note that it is not even simple to count the number of possible strategies, since some are illegal because they put more than one cup in front of a player, per-

haps in a rare configuration). Here we give some known results to give a sense of what solutions look like.

A problem in the distributed case consists of players P_1, \dots, P_m . Each P_i wants to perform a k_i/n power hour for the same global n , coordinating with the other players.

First, observe that any participant in a distributed setting can use a solo strategy and not interact with the rest of the group. Thus, if k_i is one of the possible solo cases above, this player can use that strategy if the remaining players are able to solve the smaller distributed problem. This of course includes the case that every player wants to perform a k/n power hour that is one of the possible solo cases.

With two players it is possible to perform power hours that are not possible solo, however. For example, two simultaneous $10/60$ performances are achievable as follows. Only use one shotglass. Each player does the $20/60$ strategy, transitioning \cap to \cup , and \cup to \uplus . A player receiving \uplus drinks and transitions to \cap . In each case, the single shotglass is passed to the other player, cutting the normal $20/60$ in half by splitting it evenly. \cap to \cup and place it in front of the other player. This is the power of teamwork!!

More complex arrangements are possible, like where you pass to a different dude depending on what orientation the cup is in, and who knows what happens?!

5 Future Work

5.1 Distributed Algorithms

In future research we plan to study distributed algorithms involving more than one person and/or more than one shot-glass. With multiple people cooperating during one power hour, we observe many additional possibilities for tracking the state. Assume that instead of 1 participant with one shot-glass, we have p participants with q shot-glasses among them.

5.1.1 Rotation

5.1.2 Shotglass Interactions

There are also many combinations that may result if we allow for a state to be represented by multiple (presumed indistinguishable) shotglasses. As a basic example, using two empty shotglasses, we can represent three states: $\cup\cup$, and $\cup\cap$, and $\cap\cap$.

If we allow filling one or both of the cups in the former two states, this allows for even more states, but with less possibilities to transition between states without drinking.

If we allow stacking of shotglasses, we enable even more states: $\cup\cup$, $\cap\cap$, $\cup\cap$, and $\cap\cup$. In total, this allows for seven states with two shotglasses.

This can be extended to arbitrarily high stacks, with absurd consequences. We plan to hire a set theorist to study the interactions of countably infinite and possibly even uncountably infinite stacks.

5.2 Controversy

As discussed in section 3, there are several constraints on the legitimacy of power hour algorithms. In the future we will consider potential algorithms that may result if we relax these constraints.

5.2.1 Multiple Shots per Minute

If we extend the set of possible state transitions to allow the participant to drink multiple times per minute, we enable algorithms in which $k > n$. We write $(A, \dots, Z)^m$ to denote performing the actions A through Z in sequence m times repeated.

The most basic example is to extend the classic algorithm to enable a $m * n$ power hour, as follows. On each tick, (fill, drink) m , and leave \cup .

We can also write algorithms for non-integral irregular fractional power hours. For a $m/3$ power hour (with $m \geq 1$): If \cap , flip and leave \cup . If \cup , fill and leave \uplus . If \uplus , drink, (fill, drink) $^{m-1}$, flip, leave \cap .

However, the algorithm described above provides poor load-balancing in the case of large m . We can solve this problem by distributing the multiple-drinks, as follows. (In the following description, we assume $m \cong 1 \pmod{3}$, for simplicity.) If \cap ,

flip, (fill, drink)^{(m-1)/3} and leave \cup . If \cup , fill, (drink, fill)^{(m-1)/3} and leave \uplus . If \uplus , drink, (fill, drink)^{(m-1)/3}, flip, leave \cap .

We postulate that a similar load-balancing algorithm can be applied to any existing conventional algorithm for $k \leq n$.

5.2.2 Non-Extensibility

As discussed in section 4, there are certain algorithms that provide results for k additively dependent on n . The possibilities for these are also greatly expanded given the techniques described above. One example algorithm is shown below.

If $\cap\cap$, stack and leave $\hat{\cap}$. If $\hat{\cap}$, flip the top cup and leave $\overset{\cup}{\cap}$. If $\overset{\cup}{\cap}$, flip both cups and leave $\overset{\cup}{\cup}$. If $\overset{\cup}{\cup}$, flip the top cup and leave $\overset{\cup}{\cup}$. If $\overset{\cup}{\cup}$, un-stack and leave $\cup\cup$. If $\cup\cup$, fill both glasses and drink, leaving $\cup\cup$.

With one participant using two cups, this causes a 110/60 power hour. With two participants, one drinking from each cup in the final step, this causes a 55/60 power hour.

6 Conclusion

We have presented some algorithms for k/n Power Hours, woohoo!

We wrote this paper in one hour while drinking beer.

7 Cheers

Cheers to Rob Simmons for spreading the knowledge of the original Half Power Hour formulation; to Jamie Morgenstern, Rob Arnold, and Anders "POWAH HOWAH" Schack-Nielsen for providing inspiration in the form of Power Hour Participation; to Ali Spagnola for providing musical accompaniment to our writing sprint.