Optiver puzzle solution

Calvin trip optimization

D. Josipovic

17 augustus 2017

Contents

1 Problem description

Calvin has to cross several signals when he walks from his home to school. Each of these signals operate independently. They alternate every 80 seconds between green light and red light. At each signal, there is a counter display that tells him how long it will be before the current signal light changes. Calvin has a magic wand which lets him turn a signal from red to green instantaneously. However, this wand comes with limited battery life, so he can use it only for a specified number of times.

a. If the total number of signals is 2 and Calvin can use his magic wand only once, then what is the expected waiting time at the signals when Calvin optimally walks from his home to school?
b. What if the number of signals is 3 and Calvin can use his magic wand only once?
c. Can you write a code that takes as inputs the number of signals and the number of times Calvin can use his magic wand, and outputs the expected waiting time?

2 General solution approach

The following method is used to solve the three questions. First random variables S and C are defined for each signal light and counter respectively. Then a random variable W is defined for the usage of wand which directly depends on S and C of a specific signal light n. The dependence is based on a threshold, which means that the wand will be used on red lights when the counter C exceeds certain value x. Then a random variable T is constructed in function of W, S and C which signifies the total amount of time it takes Calvin to go from home to school, given the amount of lights N and charges M available. The expected travel time $\mathrm{E}[T]$ will differ depending on the thresholds x chosen at each light. The optimal thresholds x given the amount of remaining signals n and charges m available are derived by minimizing the expected travel time: $\arg\min_x \mathrm{E}[T]$. The univariate function $\mathrm{E}[T](x)$ is quadratic and convex in region $[0, 80]$, and thus the minimum exists and is unique. The general multivariate case can be reduced to multiple univariate cases and solved recursively. This is done in Question (c) is some 10 lines of code. The solutions are verified with simulations in Appendix.

3 Definitions

N is the amount of signal lights on the road. M is the amount of charges Calvin has available at start.

Let $S_n \sim \text{Bernoulli}(\frac{1}{2})$ be the random variable for the signal light n: 1 when *red*, 0 when *green*. Note that we count the signal lights backwards: the first light Calvin encounters has index N.

Let $C_n \sim \text{Unif}(0, 80)$ be the random variable for the counter at signal light n.

Each time Calvin is in front of a signal light, he has the option to use his wand (given enough charges). Let $W_{nm} = \mathrm{I}(S_n C_n \geq x_{nm})$ be the binary random variable resulting in 1 when Calvin uses his wand on signal light n given m available charges, and otherwise 0. As the indicator function implies, we assume that Calvin uses a decision procedure at each light n based on a threshold x_{nm} which depends on the n and m, i.e. on the amount of charges he has left and the n remaining lights on his road. Other decision procedures are also possible.[1]

Let T_{NM} be the random variable for the time it takes for Calvin to go from home to school given N lights and M charges.

4 Question a: 2 lights, 1 charge

4.1 Definition of T_{21}

Logical definition of T_{21} is as follows:

[1] Decision procedure depends on information available. Here we implicitly assume that Calvin knows how many lights there are on his road. But we could assume even more. For example, Calvin might be able to see the next light, which would make x_{nm} depend on C_{n-1}, etc.

$$\begin{aligned} T_{21} := \text{if } W_{21} &: S_1C_1 \\ \text{else } (1 - W_{21}) &: S_2C_2 \end{aligned}$$

which basically says that if Calvin uses the wand on the first encountered light (i.e. $n = 2$), the total travel time will be S_1C_1 seconds (i.e. the wait time of the last signal), and if not he will be able to nullify the next light but wait S_2C_2 seconds at the current.

In mathematical form we get:

$$\begin{aligned} T_{21} &:= W_{21}S_1C_1 \\ &\quad + (1 - W_{21})S_2C_2 \\ &:= W_{21}T_{10} \\ &\quad + (1 - W_{21})S_2C_2 \\ &\quad + (1 - W_{21})T_{11} \end{aligned}$$

The second identity is written in function of the other T's. The addition of the last term will become more clear later on. Note that $T_{11} = 0$, so the last term makes no difference.

Now we need the expected value of T_{21}. Once we have that, we minimize it to get the optimal x_{21}.

4.2 Expected value of T_{21}

$$\begin{aligned} \mathrm{E}[T_{21}] &= \mathrm{E}[W_{21}]E[S_1C_1] \\ &\quad + \mathrm{E}[(1 - W_{21})S_2C_2] \end{aligned}$$

We now rewrite the parts of this equality in function of x_{21}.

4.2.1 General form for $\mathbf{E}[W]$ and $\mathbf{E}[(1 - W)SC]$

Since these forms are reusable, we write the general case. We mainly make use of law of total expectation here.

$$\begin{aligned} \mathrm{E}[W] &= \mathrm{E}[\mathrm{I}(SC >= x)] \\ &= 1 - \mathrm{E}[\mathrm{I}(SC < x)] \\ &= 1 - \mathrm{E}[\mathrm{I}(0 < x)]\,\mathrm{P}(S = 0) + \mathrm{E}[\mathrm{I}(C < x)]\,\mathrm{P}(S = 1) \\ &= 1 - \frac{1}{2} - \frac{x}{80} \cdot \frac{1}{2} \\ &= \frac{1}{2} - \frac{x}{160} \end{aligned}$$

The following solution holds only if there is dependence between W and SC:

$$\begin{aligned} \mathrm{E}[(1 - W)SC] &= \mathrm{E}[\mathrm{I}(SC < x)SC] \\ &= \mathrm{E}[\mathrm{I}(C < x)C]\,\mathrm{P}(S = 1) \\ &= \mathrm{E}[C \mid C < x]\,\mathrm{P}(\mathrm{I}(C < x) = 1)\,\mathrm{P}(S = 1) \\ &= \frac{x}{2} \cdot \frac{x}{80} \cdot \frac{1}{2} \\ &= \frac{x^2}{320} \end{aligned}$$

And finally:

$$\mathrm{E}[SC] = \mathrm{E}[S]\mathrm{E}[C] = \frac{1}{2} \cdot \frac{80}{2} = 20$$

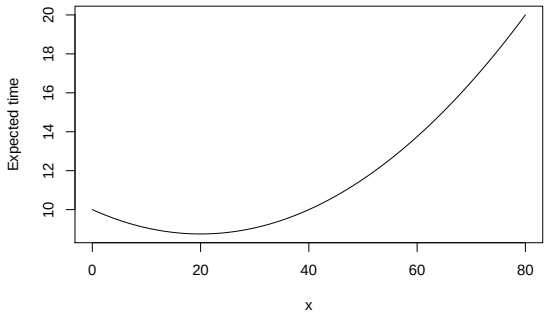
4.2.2 Solving $\arg\min_x \mathbf{E}[T_{21}]$

Thus $\mathrm{E}[T_{21}]$ becomes:

$$\mathrm{E}[T_{21}] = \frac{x^2}{320} - \frac{x}{8} + 10$$

We need the decision threshold x_{21} for which the $\mathrm{E}[T_{21}]$ is minimal, thus we solve $\arg\min_{x_{21}} \mathrm{E}[T_{21}]$:

```
local({
  etime <- function(x){x^2/320 - x/8 + 10}
  curve(etime, from = 0, to = 80, ylab = 'Expected time')
  optimize(etime, interval = c(0,80))
})
```




```
## $minimum
## [1] 20
##
## $objective
## [1] 8.75
```

Thus at first light, the optimal decision boundary to use the wand is 20 sec (i.e. if the light is red and counter $C \geq 20$ then Calvin will use his wand). With this boundary, the expected trip time is 8.75 sec.

Expected wait time at the first signal is $\mathrm{E}[(1 - W_{11})S_1C_1] = 20^2/320 = 1.25$ sec and at the second signal 8.75 - 1.25 = 7.5 sec.

5 Question b: 3 lights, 1 charge

We proceed as in the previous question by first defining T_{31}.

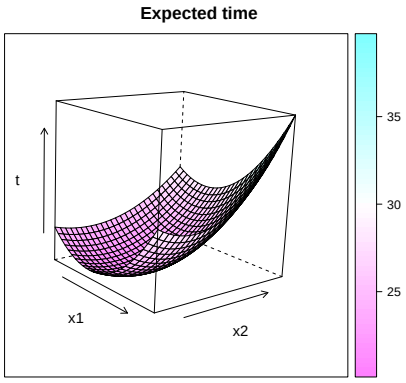
$$\begin{aligned}
T_{31} &:= \text{if } W_{31} && : S_2C_2 + S_1C_1 \\
&\quad \text{else } (1 - W_{31}) && : \text{ if } W_{21} && : S_3C_3 + S_1C_1 \\
& && \quad \text{else } (1 - W_{21}) && : S_3C_3 + S_2C_2 \\
&:= W_{31}T_{20} \\
&\quad + (1 - W_{31})S_3C_3 \\
&\quad + (1 - W_{31})T_{21}
\end{aligned}$$

T_{31} has two thresholds: x_{31} and x_{21}. Taking the expectation and minimizing it in function of x results in:

```
local({
  etime <- function(x){(1/2 - x[1]/160)*2*20 +
                       (x[1]^2/320) +
                       (1/2 + x[1]/160) * (x[2]^2/320 - x[2]/8 + 10)}
  print(optim(par = c(0,0), fn = etime, method = "L-BFGS-B"))

  # graph
  int <- seq(0,80,3)
  xy <- expand.grid(x1 = int, x2 = int)
  data <-  data.frame(x1 = xy[,'x1'], x2 = xy[,'x2'],
                      t = apply(X = xy, MARGIN = 1,
                                FUN = etime))
  lattice::wireframe(t ~ x1 * x2, data = data, drape = TRUE, colorkey = TRUE,
                     main = "Expected time",
                     screen = list(z = -60, x = -75, y = 10))
})
```

```
## $par
## [1] 31.25002 20.00001
##
## $value
## [1] 21.32324
##
## $counts
## function gradient
##        8        8
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



Note that T_{21} and thus x_{21} can be solved independently, and the result can then be used in solution of x_{31}. This fact is used in the following question where a recursive solution is made for any number of lights N and charges M.

6 Question c: N lights, M charges

We still don't know how 2 or more charges behave, so we first write T_{32} to get an idea:

$$
\begin{aligned}
T_{32} := \text{if } W_{32} \quad &: \text{if } W_{21} \quad : S_1C_1 \\
&\quad \text{else } (1-W_{21}) : S_2C_2 \\
&\text{else } (1-W_{32}) \quad : S_3C_3 \\
:= W_{32}&T_{21} \\
&+ (1-W_{32})S_3C_3 \\
&+ (1-W_{32})T_{22}
\end{aligned}
$$

The form is the same as in the previous questions and can be verified for higher N and M. Only T_{nm} is changing to T_{n-1m-1} in the first part, and to T_{n-1m} in the last part. This fact can be summarized in the following recursive function *etime(N,M)* which minimizes the expected T_{NM}.

6.1 Recursive general solution

```
etime <- function(nr.signs, nr.charges){
  EC <- 80/2 # expected count for random variable C
  ES <- 1/2 # expected sign for random variable S
  if (nr.charges == 0L) return(nr.signs * EC * ES)
  if (nr.charges >= nr.signs) return(0)
  result <- optimize(
    interval = c(0,80),
    a = etime(nr.signs = nr.signs - 1, nr.charges = nr.charges - 1),
    b = etime(nr.signs = nr.signs - 1, nr.charges = nr.charges),
    f = function(x, a, b){
      ( (1/2 - x/160) * a
        + x^2/320 # E[(1-W)CS] where CS and W are dependent
        + (1/2 + x/160) * b)
    })
  print(paste('sign:', nr.signs, '| charges:', nr.charges,
              '| optimal:', result$minimum))
  result$objective
}
```

```
etime(nr.signs = 2, nr.charges = 0)
```

```
## [1] 40
```

```
etime(nr.signs = 1, nr.charges = 1)
```

```
## [1] 0
```

```
etime(nr.signs = 3, nr.charges = 1)
```

```
## [1] "sign: 2 | charges: 1 | optimal: 20"
## [1] "sign: 3 | charges: 1 | optimal: 31.25"
```

```
## [1] 21.32324
```

```
etime(nr.signs = 4, nr.charges = 2)
```

```
## [1] "sign: 2 | charges: 1 | optimal: 20"
## [1] "sign: 3 | charges: 1 | optimal: 31.25"
## [1] "sign: 2 | charges: 1 | optimal: 20"
## [1] "sign: 3 | charges: 2 | optimal: 8.75"
## [1] "sign: 4 | charges: 2 | optimal: 17.1875"
```

```
## [1] 11.80634
```

7 Apendix

7.1 Simulation

7.1.1 Definition of objects: *Road*, *Calvin* and *Simulator*

```
Road <- function(nr.signs){

  if (nr.signs == 0) stop('nr.signs must be higher than 0')

  current.sign.nr <- 0L
  current.s <- NULL
  current.c <- NULL

  rS <- function(n = 1){
    rbinom(n, 1, 0.5)
  }

  rC <- function(n = 1){
    runif(n, min = 0, max = 80)
  }

  continue <- function(){
    if (current.sign.nr >= nr.signs) return(FALSE)
    current.sign.nr <<- current.sign.nr + 1
    current.s <<- rS()
    current.c <<- rC()
    TRUE
  }

  reset <- function(){
    current.sign.nr <<- 0
    current.s <<- NULL
    current.c <<- NULL
  }

  list(reset = reset,
       continue = continue,
       get.current.s = function(){current.s},
       get.current.c = function(){current.c},
       get.current.n = function(){nr.signs - current.sign.nr + 1})
}
```

Calvin constructor has a special function that is used internally for deciding whether wand should be used. By default, wand is used if there are enough charges.

```
Calvin <- function(nr.charges,
                   wand.decision.fn = function(...){current.nr.charges > 0}){

  current.nr.charges <- nr.charges

  use.wand <- function(){
    if (current.nr.charges < 1) stop('Can not use wand: no charges!')
    current.nr.charges <<- current.nr.charges - 1
  }

  use.wand.decision <- function(nr.signs.to.go, s, c){
    wand.decision.fn(nr.signs = nr.signs.to.go, nr.charges = current.nr.charges,
```

```
                    s = s, c = c)
  }

  list(reset = function(){current.nr.charges <<- nr.charges},
       use.wand = use.wand,
       use.wand.decision = use.wand.decision)
}
```

Simulator has the function *simulate()* which times how long it takes for Calvin to go from home to school.

```
Simulator <- function(road, calvin){

  simulate <- function(){
    road$reset()
    calvin$reset()

    ttime <- 0

    while(road$continue()){

      if (calvin$use.wand.decision(nr.signs.to.go = road$get.current.n(),
                                   s = road$get.current.s(),
                                   c = road$get.current.c())) {
        calvin$use.wand()
      } else {
        ttime <- ttime + road$get.current.s() * road$get.current.c()
      }

    }

    ttime
  }

  list(simulate = simulate)
}
```

7.1.2 Simulations

7.1.2.1 Default simulation

This is a 2-signs, 1-charge simulation with default decision rule (i.e. use if enough charges.)

```
simulator <- Simulator(road = Road(nr.signs = 2), calvin = Calvin(nr.charges = 1))
mean(replicate(n = 10000, expr = simulator$simulate()))
```

```
## [1] 20.14097
```

7.1.2.2 Optimal threshold simulation

For this simulation we use the optimal thresholds derived from the first, second and third question.

```
simulator <- Simulator(
  road = Road(nr.signs = 4),
  calvin = Calvin(nr.charges = 2,
                  wand.decision.fn = function(nr.signs, nr.charges, s, c){
                    if (nr.signs == 2L && nr.charges == 1L) return(s*c >= 20)
                    if (nr.signs == 3L && nr.charges == 1L) return(s*c >= 31.25)
                    if (nr.signs == 3L && nr.charges == 2L) return(s*c >= 8.75)
                    if (nr.signs == 4L && nr.charges == 2L) return(s*c >= 17.1875)
```

```
                    nr.charges >= 1L
                  }))
mean(replicate(n = 10000, expr = simulator$simulate()))
```

```
## [1] 11.78481
```