Produkcja na dwóch stanowiskach z kolejką

Spis treści

1 Wstęp

Tematyką drugiego projektu są symulacje z kolejkami. W tym projekcie interesuje nas praca dwóch maszyn, pomiędzy którymi znajduje się kolejka. Problemy te są interesujące, ze względu na ich rzeczywistość. Rozwiązywanie tego typu problemów jest potrzebne w każdej dziedzinie produkcji. Problemy te jednak najczęściej nie są rozwiązywalne analitycznie, więc potrzeba wykorzystać nam siły obliczeniowej komputerów, do tworzenia symulacji takich sytuacji.

2 Model (n, k)

Jak wspomniano wcześniej dysponujemy dwiema maszynami, które pracują przez 8 godzin. Zakładamy, że do pierwszej z nich sukcesywnie i natychmiastowo są donoszone elementny z magazynu, w którym znajduje się n elementów w danym dniu. Elementy te są obrabiane na maszynie, a po ich obróbce są odkładane do kolejki, która może pomieścić k elementów. Druga maszyna bierze przygotowane elementy z kolejki i dokańcza element tak, aby powstał gotowy produkt do oddania. Taki model nazywamy modelem (n, k). Dwa problemy które mogą się pojawić przy tego typu produkcji:

- kolejka jest pełena – pierwsza maszyna dokańcza element nad którym pracowała i czeka, aż druga maszyna skończy pracę nad swoim elementem, aby wziąć następny z kolejki.
- kolejka jest pusta – druga maszyna dokańcza element nad którym pracowała i czeka, aż pierwsza maszyna skończy pracę nad swoim elementem, który odda do kolejki.

Oczywistym jest, że w świcie realnym, maszyny nie będą pracować zawsze z tak samo, więc do tego modelu przyjęto czasy pracy maszyn (w godzinach):

1: $Erl(2, 10)$ (rozkład Erlanga – suma rozkładów wykładniczych).

2: $0.8Exp(9) + 0.2Exp(3)$ (mieszanka rozkładów wykładniczych).

2.1 Intuicja

Rozkład wykładniczy wydaje się być dobrym przybliżeniem realnego czasu pracy nad elementami przez maszynę, ponieważ maszyna większość elementów powinna przetworzyć szybko, z coraz mniejszym prawdopodobieństwem na dłuższą pracę. Rozkład wykładniczy ma największą masę prawdopodobieństwa na początku nośnika (krótki czas).

Rozkład $Erl(2, 10)$ jest sumą dwóch rozkładów wykładniczych $Exp(10)$ co sugerowało by wykonanie dwów akcji na pierwszej maszynie. Średnią takiego rozkładu jest równa $\frac{1}{5}$ co oznacza średnio 5 elementów na godzinę, zatem 40 przez 8 godzin pracy. Dzięki temu, wiadomym jest, że w magazynie codziennie powinno znajdować się $n > 40$ elementów, aby maszyna miała zapewnioną pracę przez cały dzień.

Mieszanka rozkładów $0.8Exp(9) + 0.2Exp(3)$ sugeruje, że większość elementów będzie obsłużona szybko, lecz czasami (z prawdp. 0.2) trafi się problem z maszyną, która będzie pracowała dłużej nad elementem niż powinna. Będzie to powodowało zapełnianie się kolejki. Częściej wybierany rozkład ma średnią $\frac{1}{9} < \frac{2}{5}$, co powoduje, że druga maszyna częściej będzie czekać, jako że szybciej pracuje od pierwszej.

Oczywistym jest teraz, że wystarczy wziąć odpowiednio duże $n > 40$ oraz $k = n$, żeby praca trwała nieustannie. Jest to jednak idealna sytuacja, w praktyce utrzymywanie kolejki jest kosztowne (przykładowo miejsce w magazynie). Dlatego, później wprowadzimy pojęcie funkcji kosztu (sek. 4.1.3), która sprowadzi to zagadaniene do ciekawszego i bardziej realistycznego.

2.2 Dane

Ze względu na znane nam warunki (założenia rozkładów) nie potrzeba prawdziwych danych do badania tego zagadnienia. Wszystkie dane będą symulowane, co jest również wygodne, ze względu na dowolną ilość replikacji takiej sytuacji. Kontrolując ilość replikacji możemy również zapewnić odpowiedni błąd z ustalonym prawdopodobieństwem.

3 Implementacja

Ze względu na tylko dwie maszyny w symulacji, istnieje prosta możliwość zaimplementowania takiej symulacji poprzez liniowe programowanie (nie jest używane programowanie równoległe, z wieloma wątkami na symulację maszyn). Maszyny są zaimplementowane jako iteratory (generatory), których następna wartość jest losowana z odpowiedniego rozkładu. Symulowany jest jeden dzień pracy, w zależności od czasu pracy, ilości elementów na magazynie n oraz długości kolejki k (dod. 1.2).

3.1 Przykładowy dzień pracy

Przykładowy 8-godzinny dzień pracy w postaci historii jest zaprezentowany w tabeli 3.1a. Zawarte w niej są odpowiednio: czas pracy, który upłynał, czas pracy pierwszej i drugiej maszyny nad elementem który jest na niej, element który znajduje się na pierwszej maszynie, elementy w kolejce oraz element który znajduje się na drugiej maszynie.

Czasy pracy i czekania maszyn z tego samego dnia zaprezentowane są w tabeli 3.1b. Zawarte w niej są odpowiednio: czas pracy maszyny pierwszej i drugiej. Gdy czas jest ujemny, oznacza to, że maszyna czeka na pracę odpowiednią ilość czasu.

Przyjrzenie się tak dobranemu (specjalnie) przykładowi potwierdza intuicje z poprzedniej sekcji (sek. 2.1). W kroku 31 pojawia się problem na maszynie drugiej, która będzie pracować nad elementem 19 prawię godzinę. To powoduje ciągłą pracę pierwszej maszyny, która zapełnia kolejkę. W kroku 35, kolejka już jest zapełniona, a maszyna kończy pracę nad elementem 25. Musi ona poczekać jednak na zwolnienie kolejki ok. 20 min. Gdy druga maszyna odblokuje się, kolejka powoli skraca się, aż do następnej podobnej stuacji.

4 Symulacje

Posiadając już odpowiednie narzędzia do symulacji, możemy zastanowić się nad analizą tego problemu. Zajmiemy się dwoma głównymi problemami: jak minimalizować czas czekania pierwszej maszyny ze względu na koszt i jak minimalizować prawdp. przestoju maszyny.

Krok	Czas pracy	Czas M1	Czas M2	Na M1	Kolejka	Na M2
1	0	0.174	0	1		0
2	0.174	0.291	0.019	2		1
3	0.465	0.052	0.333	3		2
4	0.517	0.055	0.28	4	3	2
5	0.572	0.101	0.225	5	3,4	2
6	0.673	0.38	0.124	6	3,4,5	2
7	0.797	0.256	0.014	6	4,5	3
8	0.812	0.242	0.171	6	5	4
9	0.983	0.07	0.004	6		5
10	1.053	0.296	0.115	7		6
11	1.349	0.207	0.079	8		7
12	1.556	0.132	0.207	9		8
13	1.689	0.134	0.075	10	9	8
14	1.764	0.059	0.4	10		9
15	1.822	0.185	0.342	11	10	9
16	2.007	0.082	0.157	12	10,11	9
17	2.089	0.377	0.074	13	10,11,12	9
18	2.164	0.302	0.075	13	11,12	10
19	2.238	0.228	0.115	13	12	11
20	2.353	0.113	0.221	13		12
21	2.466	0.058	0.108	14	13	12
22	2.525	0.11	0.05	15	13,14	12
23	2.574	0.06	0.007	15	14	13
24	2.582	0.052	0.006	15		14
25	2.634	0.207	0.066	16		15
26	2.841	0.141	0.041	17		16
27	2.982	0.232	0.034	18		17
28	3.214	0.029	0.281	19		18
29	3.243	0.209	0.252	20	19	18
30	3.453	0.109	0.043	21	19,20	18
31	3.496	0.066	0.994	21	20	19
32	3.562	0.171	0.928	22	20,21	19
33	3.733	0.094	0.757	23	20,21,22	19
34	3.827	0.124	0.662	24	20,21,22,23	19
35	3.951	0.222	0.538	25	20,21,22,23,24	19
36	4.489	0.196	0.035	26	21,22,23,24,25	20
37	4.525	0.161	0.127	26	22,23,24,25	21
38	4.652	0.034	0.077	26	23,24,25	22
39	4.686	0.418	0.043	27	23,24,25,26	22
40	4.729	0.375	0.028	27	24,25,26	23
41	4.756	0.347	0.284	27	25,26	24
42	5.04	0.063	0.251	27	26	25
43	5.103	0.067	0.188	28	26,27	25
44	5.171	0.25	0.12	29	26,27,28	25
45	5.291	0.13	1.593	29	27,28	26
46	5.421	0.211	1.463	30	27,28,29	26
47	5.632	0.083	1.253	31	27,28,29,30	26
48	5.714	0.256	1.17	32	27,28,29,30,31	26
49	6.884	0.148	0.25	33	28,29,30,31,32	27
50	7.135	0.296	0.002	34	29,30,31,32,33	28
51	7.137	0.294	0.348	34	30,31,32,33	29
52	7.431	0.248	0.054	35	30,31,32,33,34	29
53	7.485	0.195	0.022	35	31,32,33,34	30
54	7.506	0.173	0.047	35	32,33,34	31
55	7.553	0.126	0.078	35	33,34	32
56	7.631	0.048	0.003	35	34	33
57	7.634	0.045	0.005	35		34
58	7.679	0.339	0.078	36		35

(a) Historia 8 godzin pracy.

Krok	Czas M1	Czas M2
1	0.1739	-0.1739
2	0.2908	-0.2722
3	0.0524	0.0524
4	0.0549	0.0549
5	0.1012	0.1012
6	0.1242	0.1242
7	0.0141	0.0141
8	0.1710	0.1710
9	0.0705	-0.0663
10	0.2960	-0.1812
11	0.2074	-0.1281
12	0.1321	0.1321
13	0.0751	0.0751
14	0.0585	0.0585
15	0.1848	0.1848
16	0.0824	0.0824
17	0.0744	0.0744
18	0.0746	0.0746
19	0.1149	0.1149
20	0.1129	0.1129
21	0.0584	0.0584
22	0.0499	0.0499
23	0.0074	0.0074
24	0.0523	-0.0458
25	0.2070	-0.1413
26	0.1413	-0.1001
27	0.2320	-0.1981
28	0.0289	0.0289
29	0.2094	0.2094
30	0.0429	0.0429
31	0.0661	0.0661
32	0.1708	0.1708
33	0.0944	0.0944
34	0.1241	0.1241
35	-0.3167	0.5383
36	0.0355	0.0355
37	0.1272	0.1272
38	0.0336	0.0336
39	0.0430	0.0430
40	0.0279	0.0279
41	0.2839	0.2839
42	0.0630	0.0630
43	0.0675	0.0675
44	0.1204	0.1204
45	0.1299	0.1299
46	0.2107	0.2107
47	0.0826	0.0826
48	-0.9143	1.1701
49	-0.1026	0.2502
50	0.0021	0.0021
51	0.2942	0.2942
52	0.0536	0.0536
53	0.0217	0.0217
54	0.0470	0.0470
55	0.0777	0.0777
56	0.0033	0.0033
57	0.0448	-0.0393
58	0.3388	-0.2605

(b) Czasy pracy i czekania maszyn.

Tablica 3.1: Przykładowy 8-godzinny dzień pracy.

4.1 Problem kosztu przestoju maszyny

W tym problemie zakładamy, że przestój pierwszej maszyny jest kosztowny co jest naturalnym założeniem. Oczywiście jak wspomniano wcześniej, idealnym rozwiązaniem tego problemu jest ustawienie kolejki równej liczbie wszystkich elementów w magazynie. Jest to już jednak nierealne rozwiązanie, ze względu na koszt kolejki.

4.1.1 Frakcja czasu niepracowania

Do tego problemu posłużymy się zmienną losową f_1, zależną od k, która będzie oznaczać frakcję czasu nieprawcowania pierwszej maszyny w ciągu całego dnia, definiowaną tak:

$$t_i = ((\text{Czas M1})_i)_-, \quad f_1 = \frac{1}{8h}\sum_{i=1}^{N} t_i$$

Gdzie $i = 1, \ldots, N$, a N to ilość kroków w symulacji jednego dnia. Oczywiście przy $n = k$ mamy $f_1 = 0$, zatem $f_1 \overset{k\to n}{\to} 0$. Rozkład tej zmiennej nie jest nam znany, więc jej średnia będzie symulowana poprzez *crude monte carlo estimator* (dod. 1.3):

$$\hat{Ef_1} = \frac{1}{R}\sum_{i=1}^{R} f_{1i}$$

Gdzie R to liczba replikacji, f_{1i} to i-ta (niezależna od innych) replikacja zmiennej f_1.

4.1.2 Liczba replikacji

Do zapewnienia odpowiedniego błedu przy ustalonym prawdp. potrzebne jest znać liczbę replikacji. Gdy $Y = f_1$ to możemy skorzystać z fundamentalnego związku:

$$b = \frac{z_{1-\alpha/2} \cdot \sigma_Y}{\sqrt{R}} \tag{1}$$

Gdzie $z_{1-\alpha/2}$ jest kwantylem rzędu $1 - \alpha/2$ standardowego rozkładu normalnego, σ_Y jest odchyleniem standardowym zmiennej Y, R liczbą replikacji, b oczekiwanym błędem. Przy ustalonym błędzie oraz znanej wariancji, można obliczyć liczbę replikacji do utrzymania błędu b z prawdp. $1 - \alpha$.

Niestety w związku z brakiem wiedzy o rozkładzie f_1 musimy symulować wariancję (dod. 1.4), za pomocą nieobciążonego estymatora wariancji, do którego zostanie użyte 10^4 replikacji. Przy $\alpha = 0.1$ wyniki tej symulacji zaprezentowane są w tabelach 4.1 i 4.2. Do symulacji przyjęto różne błędy, ze względu na dążącą do zera wartość oczekiwaną zmiennej f_1 (błedy powinny być coraz mniejsze, żeby miały sens).

k	1	2	3	4	5	6	7	8	9	10
b	0.01	0.01	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
R	235	205	16585	13639	10788	8078	6207	4723	3372	2423

Tablica 4.1: Liczba replikacji zm. f_1 dla $k = 1, \ldots, 10$.

k	11	12	13	14	15	16	17	18	19	20
b	0.001	0.001	0.001	0.0001	0.0001	0.0001	0.0001	0.0001	0.00001	0.00001
R	1609	1124	773	58854	32438	32444	17919	11592	1291818	461439

Tablica 4.2: Liczba replikacji zm. f_1 dla $k = 11, \ldots, 20$.

Gdy k zbliża się do 20, prawdp. przestoju jest już na tyle bliskie zeru, że błąd musi być zbliżenie mały, co powoduje mnożenie (oraz druga potęga błedu) przez duże potęgi 10. Wariancja również maleje, jednak nie na tyle szybko, żeby znacznie zmniejszyć to mnożenie, co powoduje tak ogromne liczby replikacji. Z praktycznych powodów (czas obliczeń), przyjęto, aby obciąć ilość replikacji do maksymalnie 10^4, ponieważ do odpowiedzi na następne pytania, $k < 20$ jest wystarczające – prawdp. pustej kolejki przy $k > 18$ jest praktycznie zerowe.

4.1.3 Funkcja kosztu

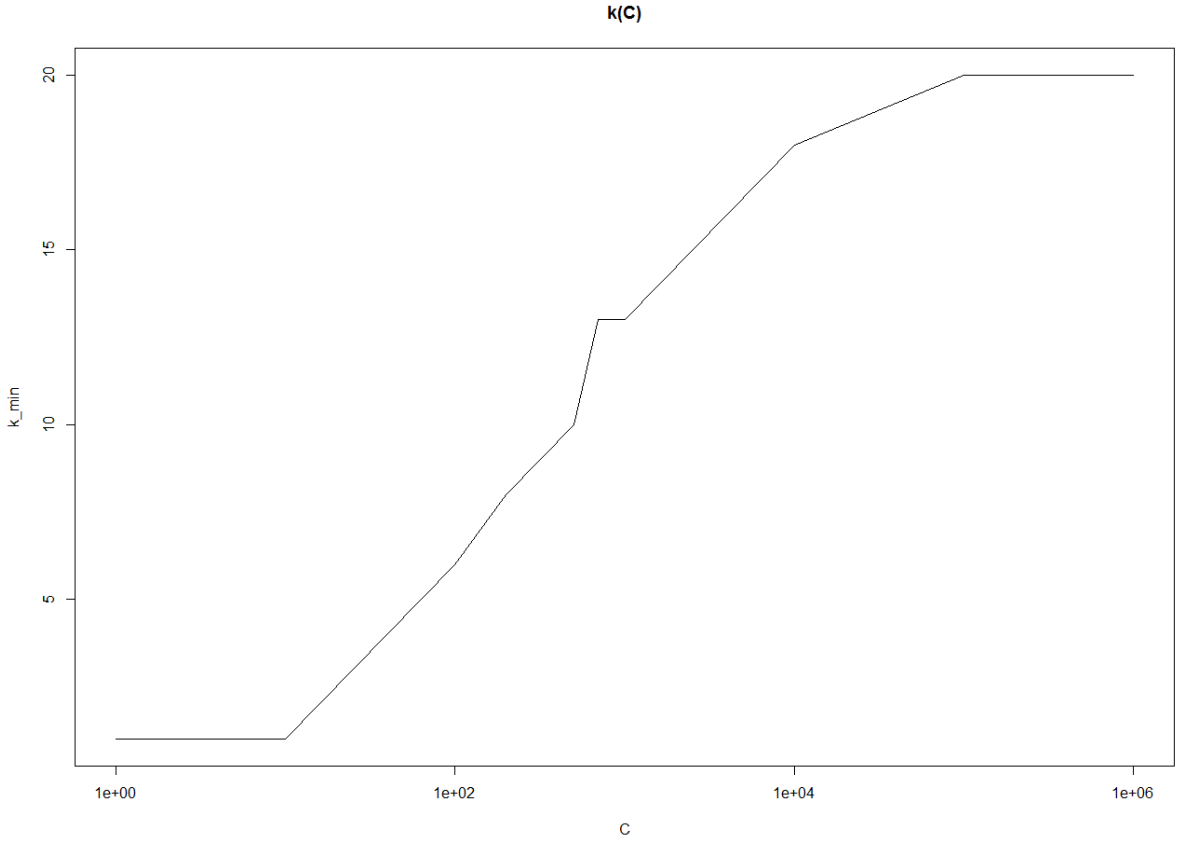
Korzystając ze zmiennej zdefiniowanej w poprzedniej sekcji (sek. 4.1.1) możemy zdefiniować funkcję kosztu, która będzie zależeć od k (dod. 1.5):

$$c(C, k) = CEf_1 + k$$

Wiadomym jest, że średnia zm. f_1 jest malejąca przy rosnącym k. Oznacza to, że przy zwiększaniu kolejki (k) pierwszy składnik w funkcji kosztu będzie malał, jednak drugi będzie rósł. Z tego powodu, można podejrzewać, że funkcja kosztu będzie posiadała minimum (przy ustalonym C).

4.1.4 Minimalna kolejka dla funkcji kosztu

Ustalając pewne C możemy znaleźć numerycznie dla funkcji kosztu minimum w zależności od k (dod. 1.6). W ten sposób możemy zdefiniować funkcję $k_0(C)$, której wartości to k minimalizujące funkcję kosztu dla C (dod. 1.7). Wykres funkcji $k_0(C)$ jest przedstawiony na rysunku (4.1). Oś OX (zmiennej C) jest w skali **logarytmicznej**.



Rysunek 4.1: Wykres funkcji $k_0(C)$.

Wybrane punkty do narysowania wykresu to $C = 10^0, 10^1, 10^2, 2 \cdot 10^2, 5 \cdot 10^2, 7 \cdot 10^2, 10^3, 10^4, 10^5, 10^6$. Gdy stała C jest mała, w koszcie dominuje wtedy liniowy składnik k. Oczywistym dlatego jest, że dla mniejszych C wystarczy kolejka o długości 1. Ef_1 jest liczbą zawsze bliską zeru (dla $k = 1$, $\hat{Ef_1} = 0.16$), więc potrzeba wysokiego C, aby ten składnik przewyższył liniowy składnik. Gdy C jest rzędu setek tysięcy lub milionów, musimy to zrekompensować na tyle dużą kolejką ($k = 18, 19, 20$), by średnia czekania była bliska zeru, żeby jej składnik zaniknął.

4.2 Problem wielkości kolejki

W tym problemie jesteśmy zainteresowani znalazieniem minimalnej długości kolejki (k), przy której wiemy, że z prawdop. większym niż 0.9 praca na pierwszej maszynie nie ustanie, co przekłada się na warunek $P(f_1 = 0) \geq 0.9$.

4.2.1 Brak przestoju maszyny

Zdefiniujmy zmienną 0/1 odpowiadającą braku przestoju w pracy pierwszej maszyny:

$$B = \begin{cases} 1 & \text{gdy } f_1 = 0 \\ 0 & \text{gdy } f_1 \neq 0 \end{cases}$$

Taka zmienna jest zmienną z rozkładu dwumianowego $b(1, p)$, gdzie $p = P(f_1 = 0)$. Wartość oczekiwana zmiennej B wyniosi:

$$EB = 0 \cdot P(f_1 \neq 0) + 1 \cdot P(f_1 = 0) = p$$

Oznacza to, że nasze poszukiwane prawdp. można estymować za pomocą *crude monte carlo estimator* zmiennej B, który ma postać:

$$\hat{EB} = \frac{1}{R} \sum_{i=1}^{R} B_i$$

Gdzie R to liczba replikacji, B_i to i-ta (niezależna od innych) replikacja zmiennej B.

4.2.2 Liczba replikacji

Podobnie jak w sekcji 4.1.2 należy dobrać odpowiednią ilość replikacji, korzystając z fundamentalnego związku (1) do trzymania błędu z ustalonym prawpd. Tym razem jednak możemy skorzystać z faktu, że znamy wzór na wariancję zmiennej B: $VarB = p(1 - p)$. Jest to funkcja kwadratowa zmiennej p, z ramionami skierowanymi w dół, której maksimum jest w $\frac{1}{2}$ i wynosi $\frac{1}{4}$. Zatem możemy oszacować od góry wariancję przez $\frac{1}{4}$. Wtedy z fundamentalnego związku (przy $b = 0.01$ i $\alpha = 0.1$):

$$R = \frac{z^2_{1-\alpha/2} \cdot \sigma^2_B}{b^2} \leq \frac{z^2_{1-\alpha/2}}{4b^2} = 6763.859$$

Chcąc jednak przyspieszyć czas obliczeń, liczba replikacji (dla $b = 0.01$ i $\alpha = 0.1$) została obliczona poprzez szacowanie wariancji jak w sekcji 4.1.2 (dod. 1.8) i przedstawiona w tabelach 4.3 i 4.4. Możemy

k	1	2	3	4	5	6	7	8	9	10
R	143	1739	4570	6327	6760	6299	5157	4235	3393	2412

Tablica 4.3: Liczba replikacji zm. B dla $k = 1, \ldots, 10$.

k	11	12	13	14	15	16	17	18	19	20
R	1889	1325	1025	637	562	356	234	127	106	73

Tablica 4.4: Liczba replikacji zm. B dla $k = 11, \ldots, 20$.

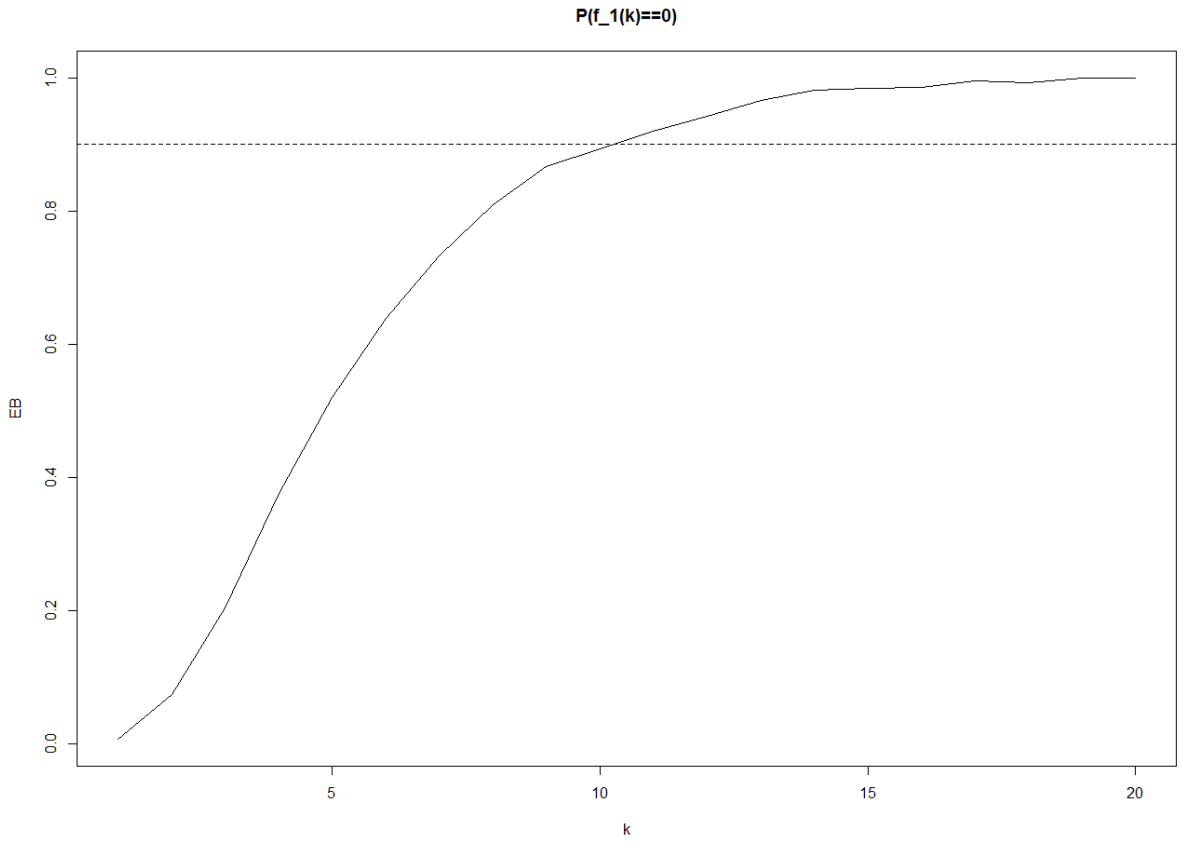
zauważyć, że rzeczywiście ilość replikacji sięga ograniczenia wyliczonego wcześniej, lecz w niektórych przypadkach, możem oszczędzić wielu replikacji.

4.2.3 Minimalna kolejka dla braku przestoju

Estymmator $\hat{EB}$ jest zależny od k, zatem można go przedstawić w postaci wykresu od zmiennego k, który jest zademonstrowany na rysunku 4.2 (dod. 1.9). Funkcja $\hat{EB}(k)$ przecina się z poziomą prostą na poziomie 0.9 dla k pomiędzy 10, a 11 – $\hat{EB}(10) = 0.89$ natomiast $\hat{EB}(11) = 0.92$. Z błedem 0.01 można oczekiwać, że dla $k = 10$ prawdopodobieństwo przestoju maszyny może być równe 0.9, a z pewnością 90% można powiedzieć, że dla $k = 11$ maszyna nie stanie.

5 Wnioski

Wszystkie obliczenia były replikowane odpowiednio tyle razy, żeby z prawdp. 90% utrzymać odpowiedni błąd (w przypadku drugiego problemu $b = 0.01$). Zatem na każde pytanie, możemy odpowiedzieć z 90% pewnością.



Rysunek 4.2: Wykres $\hat{EB}(k)$, pozioma linia na poziomie 0.9.

Pewne intuicje zostały potwierdzone – dla małych kolejek czasy czekania są duże, oraz istnieje większe prawdp. przestoju maszyny. Przy małych C minima funkcji kosztu pozostają przy 1, ponieważ składnik liniowy k jest wtedy silniejszy (średnie czasy czekania są i tak małe), natomiast dla dużych C zależy nam na zmniejszeniu czekania, zatem minimalne k zwiększa się.

Z pewnością 90% wiemy, że dla kolejki o długości $k = 11$ mamy prawpd. przestoju pierwszej maszyny 0.9. Taka kolejka prawdopodobnie jest blisko minimum kosztu gdy $C \in [500, 700]$.

Z sekcji o liczbach replikacji (sek. 4.1.2) możemy zauważyć istotę symulacji. Największa ilość replikacji w tych eksperymentach, to 10^4, co przekładało by się w świecie realnym na 27 lat codziennej produkcji, a komputerowi zajmuje od kilkunastu minut do godziny. Nawet jeżeli jest to możliwe technicznie, to w tym czasie, technologia może zmienić się, a dane mogą stać się bezużyteczne. Symulacje tego typu pozwalają nam, na odpowiedzenie na różne pytania potrzebne aktualnie, z porządaną pewnością i błedem.

Appendices

Dodatek 1

Listing 1.1: Maszyny i ich funkcje.

```
M1 = iter(function() sum(rexp(2,10)))
M2 = iter(function() ifelse(rbinom(1,1,.8), rexp(1,9), rexp(1,3)))

enqueue <- function(queue, element) c(queue, element)
dequeue <- function(queue) queue[-1]
```

Listing 1.2: Symulacja dnia pracy.

```
production <- function(work_time=8, n=120, k=5, machine1=M1, machine2=M2) {
  time = 0; i=1; product = 1; on_m1 = 1; on_m2 = 0;
  buffer = NULL; time_1 = NULL; time_2 = NULL;
  history = data.frame()

  t1 = nextElem(machine1); t2 = 0;
  history = rbind(history, c(round(time,3), round(t1,3), round(t2,3), on_m1
      , paste(buffer, collapse = ","), on_m2))

  while (time < work_time & on_m2 <= n) {
    waiting = length(buffer)

    if(waiting == 0){ #IS BUFFER EMPTY?
      time = time + t1
      time_1[i] = t1 #producing product
      buffer = enqueue(buffer, on_m1) #enqueuing product
      on_m1 = on_m1 + 1 #next product on machine

      if(t2 - t1 < 0){ #2ND MACHINE WORKING PARALLEL
        time_2[i] = t2 - t1 #done and waiting for product
        on_m2 = buffer[1]
        buffer = dequeue(buffer) #removing from buffer right after
            enqueuing
        t2 = nextElem(machine2)
      }
      else{
        t2 = t2 - t1 #working on product on 2nd machine
        time_2[i] = t1
      }

      t1 = nextElem(machine1)
    }
      else if(waiting == k){ #IS BUFFER FULL?
        time = time + t2
        time_2[i] = t2 #working on product
        on_m2 = buffer[1]
        buffer = dequeue(buffer)

        if(t1 - t2 < 0){ #1ST MACHINE WORKING PARALLEL
          time_1[i] = t1 - t2 #done and waiting for space
          buffer = enqueue(buffer, on_m1) #enqueuing right after removing
          on_m1 = on_m1 + 1 #next product on machine
          t1 = nextElem(machine1)
        }
```

```
      else{
        t1 = t1 - t2 #producing the product on 1st machine
        time_1[i] = t2
      }

      t2 = nextElem(machine2)
    }
      else if(t1<t2){ #WHO'S FASTER?
        time = time + t1
        t2 = t2 - t1
        time_1[i] = t1
        time_2[i] = t1
        buffer = enqueue(buffer, on_m1)
        on_m1 = on_m1 + 1
        t1 = nextElem(machine1)
      }
      else{
        time = time + t2
        t1 = t1 - t2
        time_1[i] = t2
        time_2[i] = t2
        on_m2 = buffer[1]
        buffer = dequeue(buffer)
        t2 = nextElem(machine2)
      }


    history = rbind(history, c(round(time,3), round(t1,3), round(t2,3), on_
        m1, paste(buffer, collapse = ","), on_m2))
    i = i+1
  }
  colnames(history) = c("time", "t1", "t2", "on_m1", "buffer", "on_m2")
  return(list(history=history, times=cbind(time_1, time_2)))
}
```

Listing 1.3: Symulacja zmiennej f_1.

```
f_1 <- function(rep=100, work_time=8, n=120, k=5, machine1=M1, machine2=M2)
    {
  f = NULL
  for (i in 1:rep) {
    time1 = production(work_time, n, k, machine1, machine2)$times[,1]
    f[i] = sum(abs(time1[time1<0])) / work_time#/ length(time1)
  }
  return(f)
```

Listing 1.4: Własności f_1.

```
sigma_f_1 <- function(rep=1000, work_time=8, n=120, k=5, machine1=M1,
    machine2=M2) {
  return(var(f_1(rep, work_time, n, k, machine1, machine2)))
}

#How many replications are needed? k=1,5,10 -> R=400,100,20
R <- function(b=.01, alpha=.1, rep=10000, work_time=8, n=120, k=5, machine1
    =M1, machine2=M2) {
  return(ceiling(sigma_f_1(rep, work_time, n, k, machine1, machine2) *
      qnorm(1-alpha/2)^2 / b^2))
}
```

Listing 1.5: Funkcja kosztu.

```
cost <- function(C, k, Ef_1){
  return(C * Ef_1 + k)
}
```

Listing 1.6: Szukanie minimalnego k dla kosztu.

```
k_min <- function(C, Ef_1, K=1:20) {
  costs = cost(C, K, Ef_1)
  return(which.min(costs))
}
```

Listing 1.7: Funkcja $k_0(C)$.

```
reps_trim = ifelse(reps>10^4, 10^4, reps)
Ef_1 = NULL
for (k in 1:20) {
  Ef_1[k] = mean(f_1(rep = reps_trim[k], k=k))
  cat("k= ", k, "rep=",rep[k],"Ef_1=", Ef_1[k],"\n")
}
analyze_k_min <- function(costs=c(10^(0:2), c(2,5,7)*10^2, 10^(3:6)), Ef_1)
    {
  k_mins = NULL;
  for (i in 1:length(costs)) {
    C = costs[i]
    k_mins[i] = k_min(C, Ef_1)
    cat("C= ", C, "k_min= ", k_mins[i], "\n")
  }
  plot(costs, k_mins, main = "k(C)", log = "x", type = "s", xlab = "C",
      ylab = "k_min")
  return(cbind(costs, k_mins))
}
test = analyze_k_min(Ef_1 = Ef_1)
```

Listing 1.8: Własności zmiennej B.

```
sigma_f_1_p0 <- function(rep=1000, work_time=8, n=120, k=5, machine1=M1,
    machine2=M2) {
  return(var(f_1(rep, work_time, n, k, machine1, machine2) == 0))
}

#How many replications are needed? k=1,5,10 -> R=400,100,20
R_p0 <- function(b=.01, alpha=.1, rep=10000, work_time=8, n=120, k=5,
    machine1=M1, machine2=M2) {
  return(ceiling(sigma_f_1(rep, work_time, n, k, machine1, machine2) *
      qnorm(1-alpha/2)^2 / b^2))
}

Ef_1_p0 = NULL
for (k in 1:20) {
  Ef_1_p0[k] = estimate_f_1_p0(rep=1000, k=k)
  cat("k= ", k,"Ef_1_p0=", Ef_1_p0[k],"\n")
}

reps_p0 = NULL
for(k in 1:20){
  reps_p0[k] = R_p0(k=k)
  cat("k= ",k, "rep= ",reps_p0[k],"\n")
}
```

```
#Probability that f_1 == 0
estimate_f_1_p0 <- function(rep=100, work_time=8, n=120, k=5, machine1=M1,
    machine2=M2) {
  return(mean(f_1(rep, work_time, n, k, machine1, machine2) == 0))
}
```

Listing 1.9: Funkcja $EB(k)$.

```
analyze_k_min_p0 <- function(reps=reps_p0, work_time=8, n=120, machine1=M1,
    machine2=M2) {
  est = NULL
  for (k in 1:20) {
    est[k] = estimate_f_1_p0(reps[k], work_time=8, n=120, k=k, machine1=M1,
        machine2=M2)
    cat("est= ", est[k], "k= ", k, "\n")
  }
  #return(min(which(est>=0.9)))
  plot(1:20, est, type = "l", main = "P(f_1(k)==0)", xlab="k", ylab = "EB")
  abline(h=0.9, lty="dashed")
  return(est)
}
```