Recherche opérationnelle

Simulation d'un système de file d'attente

Aline Barbiaux – Céline Delhaye – Manon Libert

IG3 – 2016-2017

Enoncé

Un système d'attente se caractérise par

- ***une loi des arrivées des clients :***

Arrivées par minute	0	1	2	3	4	5
Répétitions	5	2	3	28	12	7

- ***une loi des services :***

Durée en minutes	1	2	3	4	5	6
Répétitions	2	3	2	11	17	22

Les hypothèses suivantes sont établies :

- Il y a trois chances sur 10 pour qu'un client entrant soit prioritaire absolu et une chance sur 10 pour qu'il soit prioritaire relatif.

- Il y a trois files. La capacité de la file des prioritaires absolus est de 5 clients. Dès lors, si un prioritaire absolu doit se mettre en 6ème position, il devient prioritaire relatif. Les files des prioritaires relatifs et des clients ordinaires sont illimitées.

- Le temps de simulation est de 960 minutes.

- Dans chaque file, les temps de services les plus courts sont privilégiés.

- Un prioritaire absolu éjecte un ordinaire en service si aucun autre service n'est disponible. Ce sera le client qui a la plus longue durée de service restante. Dans ce cas, il devient prioritaire relatif.

Les coûts unitaires sont :

- pour une heure de présence dans le système : prioritaire absolu : 37,5 euros, prioritaire relatif : 25,5, ordinaire : 22,5 ;

- pour une heure d'occupation d'une station : 30 ;

- pour une heure d'inoccupation d'une station : 18. Déterminez le nombre optimal de stations à ouvrir en régime continu.

Calculs de base

Obtention de Smin.

Arrivé des clients:

$\mathbf{xi} * \mathbf{ri} = \mathbf{175}$ $\mathbf{N} = \mathbf{57}$.

$\boldsymbol{\lambda} = \frac{\mathbf{175}}{\mathbf{57}} = \mathbf{3,0702}$

Durée de service:

$xi * ri = 275$ N = 57

$\mu = \frac{1}{4{,}8245}$

$\varphi = 3{,}0702 * 4{,}8245$

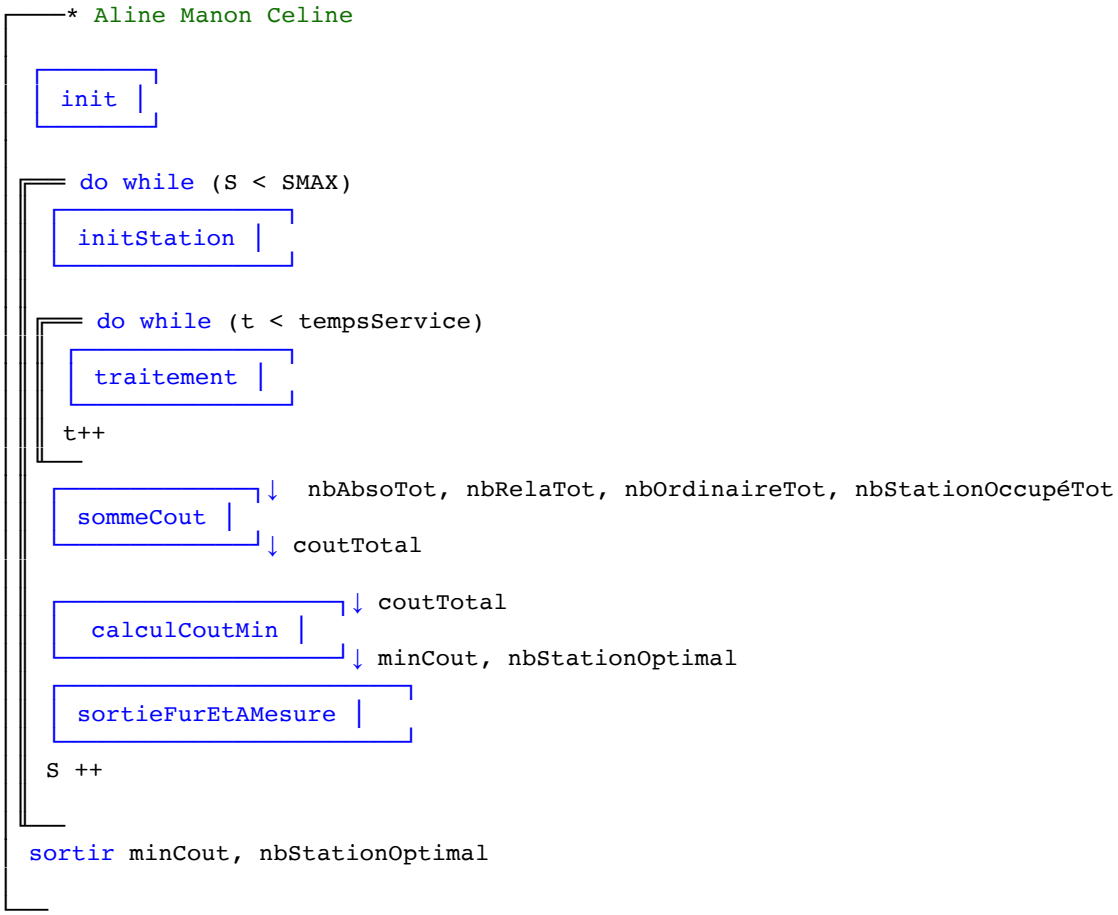
$= 14{,}81$

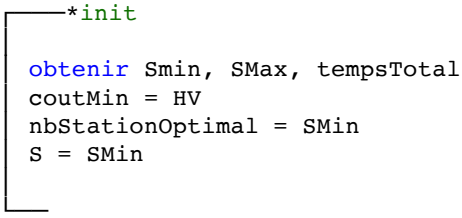
Le minimum de station sera donc de 15. Ce qui signifie qu'a 15 stations le système sera 98% occupé.

Obtention de Smax

Sachant que nous aurons au maximum 6min de durée de service et 5 arrivé par minute on considère que Smax = 5 * 6 = 30.

Diagramme d'action





* initStation

```
ptrDebOrdinaire = ptrDebAbsolu = ptrDebRelatif = NULL

t = 0
```

init tabStation

```
nbAbsoluTot = nbrelati.fTot = nbOrdinaireTot = nbStationTot = 0
cptSuiteAleatoire = 0
xn = xo
```

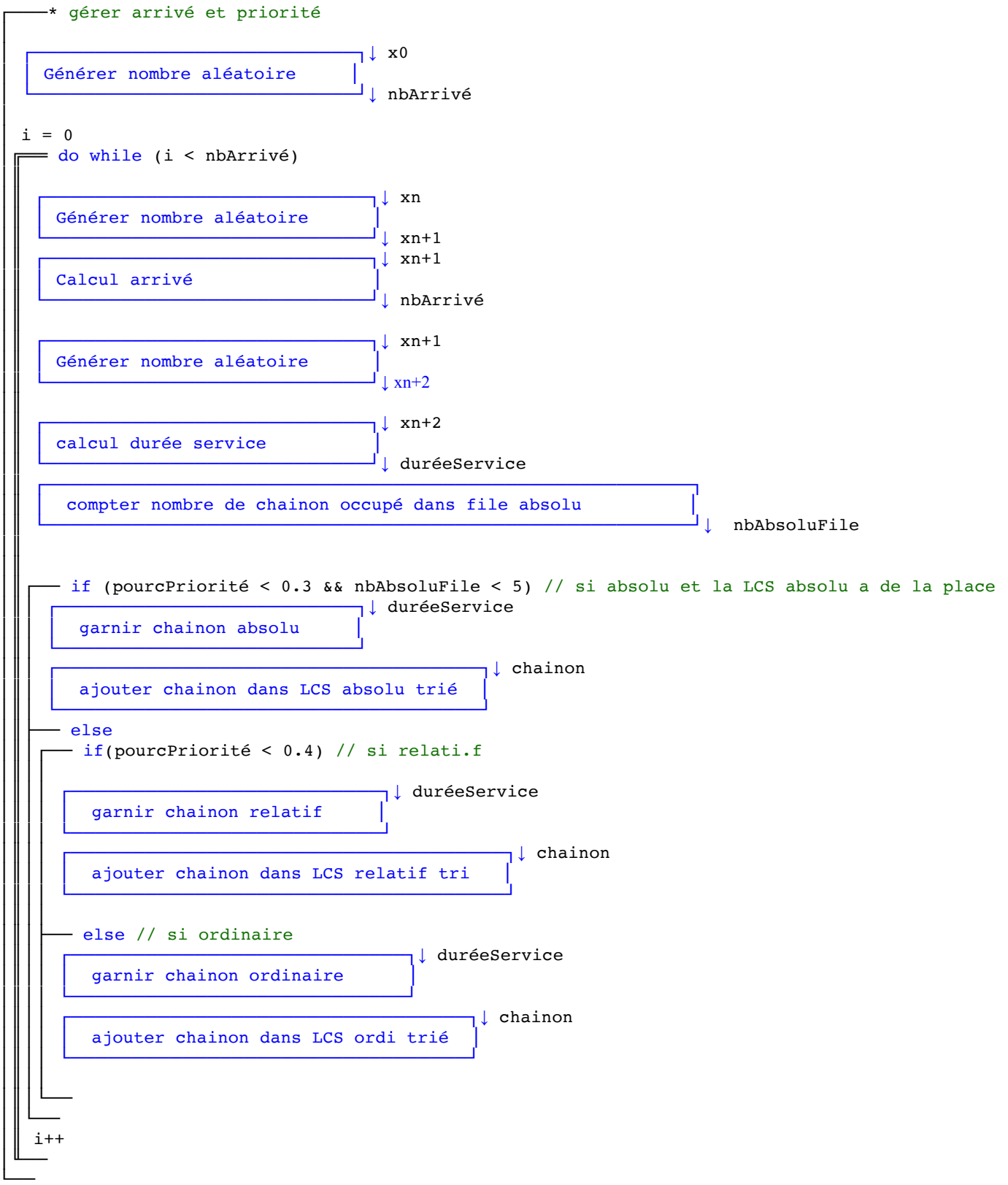
*Traitement

gérer arrivé et priorité

gérer stations libre

gérer stations occupé

Calcule des totaux


* gérer arrivé et priorité
Générer nombre aléatoire
x0
nbArrivé
i = 0
do while (i < nbArrivé)
Générer nombre aléatoire
xn
xn+1
Calcul arrivé
xn+1
nbArrivé
Générer nombre aléatoire
xn+1
xn+2
calcul durée service
xn+2
duréeService
compter nombre de chainon occupé dans file absolu
nbAbsoluFile
if (pourcPriorité < 0.3 && nbAbsoluFile < 5) // si absolu et la LCS absolu a de la place
garnir chainon absolu
duréeService
ajouter chainon dans LCS absolu trié
chainon
else
if(pourcPriorité < 0.4) // si relati.f
garnir chainon relatif
duréeService
ajouter chainon dans LCS relatif tri
chainon
else // si ordinaire
garnir chainon ordinaire
duréeService
ajouter chainon dans LCS ordi trié
chainon
i++

```
*Générer nombre aléatoire

x = (A*xn + C) % M;
```

```
*gérer stations libre
iStation = 0
nbOrdTraité = 0

do while (iStation < S)

 if (tabStation(iStation).dureeService ≤ 0) //si la station est libre
  if(ptrDebAbsolu ≠ NULL) //Et qu'il y a des absolu
  tabStation(iStation).dureeService = pDebAbso→dureeService
  tabStation(iStation).prio = 1
                                  ↓ ptrDebAbsolu
   suppressionChainon

  else
   if(ptrDebRelati.f ≠ NULL)
   tabStation(iStation).dureeService = pDebRejet→dureeService
   tabStation(iStation).prio = 2
                                  ↓ ptrDebRelatif
    suppressionChainon

   else
    if(ptrDebOrd ≠ NULL)
    tabStation(iStation).dureeService = pDebOrd→dureeService
    tabStation(iStation).prio = 3
                                  ↓ ptrDebOrdi
     suppressionChainon

 if(tabStation(iStation).dureeService > 0)

  if (tabStation(iStation).prio = 3)
  nbrOrdTraité++ //on aura le nombre total ordinaire au moment t


 iStation++
```

```
* gérer stations occupés

do while (pDebAbso ≠ 0 AND nbOrdTraité ≠ 0) //tant qu'il y a des absolus en attente
//et de ordinaires en traitement
iStation = 0
maxTemps = 0
   do while (iStation < S)
      if (tabStation(iStation).prio = 3)
       if(tabStation(iStation).dureeService > maxTemps)
        maxTemps = tabStation(iStation).dureeService
        numStation = iStation
      iStation++

pNouvRelatlf = nouveau chaînon relatlf

pNouvRelatlf→dureeService = maxTemps
o──────────────────────────o↓ pNouvRelatif,pDebRelatif
| AjouterListeRelative tri |
o──────────────────────────o
tabStation(numStation).dureeService = pDebAbso→dureeService
tabStation(numStation).prio = 1
                           ↓ pDebAbso
 suppressionChainon        |

nbOrdTraité--
```

```
* calcul des totaux

iStation = 0

do while (iStation < S)

 if (tabStation(iStation).duréeService > 0)
  nbStationTot ++
  // on aura a chaque tout le nombre de station occupé
  // pourra être un case
  if (tabStation(iStation).prio = 1)
  nbAbsoluTot++

  if (tabStation(iStation).prio = 2)
  nbrelati.fTot++

  if (tabStation(iStation).prio = 3)
  nbOrdinaireTot++

  tabStation(iStation).duréeService -- // on va passer a la prochaine minute


 iStation++


ptr = ptrDebAbso

do while (ptr ≠ NULL)
 nbAbsoluTot++
 ptr = ptr→Suiv


ptr = ptrDebRelati.f

do while (ptr ≠ NULL)
 nbRelati.fTot++
 ptr = ptr→Suiv


ptr = ptrDebOrdinaire

do while (ptr ≠ NULL)
 nbOrdinaireTot++
 ptr = ptr→Suiv
```

```
*SommeCout
totOrdinaires = nbOrdinaires * COUT_ORDINAIRE / 60;
totAbsolus = nbAbsolus * COUT_ABSOLU / 60;
totRelatives = nbRelatives * COUT_RELATIF / 60;
stationsLibres = station * TEMPS_SERVICE - nbStationsTotal;
coutStationsLibres = stationsLibres * COUT_STATION_LIBRE / 60;
coutStationsOccupes = nbStationsTotal * COUT_STATION_OCCUPE / 60;
```

```
*Calcul durée service
pourcDurée = nbPseudoAleatoire / M;
if (pourcDurée < 0.04)
return 1
else
if (pourcDurée < 0.09)
return 2
else
if (pourcDurée < 0.12)
return 3
else
if (pourcDurée < 0.32)
return 4
else
if (pourcDurée < 0.61)
return 5
else
return 6
```

```
*Calcul arrivée
pourcArrivé = nbPseudoAleatoire / M;
 if (pourcArrive < 0.09)

 return 0

 else

  if (pourcArrive < 0.12)

  return 1

  else

   if (pourcArrive < 0.18)

   return 2

   else

    if (pourcArrive < 0.66)

    return 3

    else
     if (pourcArrive < 0.88)

     return 4

     else

     return 5
```

Conclusions de fin du dossier

Nous pouvons arriver à la conclusion que le nombre optimal de stations ouverte est 17.
Quelques chiffres:
nombre de station = 17

cout pour les ordinaires : 3604.875000
cout pour les relatifs : 609.875000
cout pour les absolus : 2538.125000
cout pour les stations occupÈes : 7235.000000
cout pour les stations libres : 554.700000
cout total = 14542.575000