
Laboratoire 1 : espérance de vie d'un système

Conditions particulières :

- ▷ Suivre les recommandations et respecter les contraintes énoncées dans les conditions générales distribuées.
 - ▷ À rendre au plus tard le lundi 20 décembre 2004 à 17h.
 - ▷ L'archive Labo1.tgz est disponible sur <http://ina.eivd.ch/collaborateurs/etr/msd>
 - ▷ Seuls fichiers à rendre : `generateurs.c`, `reseau.c`, `simulation.c`, `statistiques.c` et `echeancier.h`.
-

Contexte :

Une préoccupation de la théorie de la **fiabilité** est de calculer l'**espérance de vie** d'un **système** dont les **composantes** sont non seulement sujettes à des **pannes**, mais également soumises à une certaine politique de **maintenance**. Dans cette étude, nous nous limiterons à des systèmes dont les constituants sont soumis à des pannes **cataleptiques**, c'est-à-dire à des altérations soudaines et complètes de leur fonctionnement (par opposition aux pannes **par dérive** qui découlent de dégradations lentes et progressives des performances, généralement par usure naturelle).

On modélise un **système** à l'aide d'un couple (E, φ) où

- ▷ E désigne l'ensemble de ses k **composantes** : $E = \{e_1, \dots, e_k\}$,
- ▷ $\varphi : \{0, 1\}^k \rightarrow \{0, 1\}$ est sa **fonction structurelle** associée.

Plus précisément, à chaque instant $t \in \mathbb{R}_+$ et pour tout $i \in \{1, \dots, k\}$, on associe une variable binaire $x_{i,t} \in \{0, 1\}$ décrivant l'**état de fonctionnement** de la composante e_i à l'instant t

$$x_{i,t} = \begin{cases} 1 & \text{si la composante } e_i \text{ fonctionne au temps } t, \\ 0 & \text{sinon.} \end{cases}$$

Par ailleurs, la fonction structurelle φ associe au vecteur-ligne $\vec{x}_t = (x_{1,t}, \dots, x_{k,t}) \in \{0, 1\}^k$, qui décrit l'état simultané des k composantes au temps t , une variable binaire $\varphi(\vec{x}_t)$ qui indique si globalement le système fonctionne au temps t :

$$\varphi(\vec{x}_t) = \begin{cases} 1 & \text{si le système fonctionne au temps } t, \\ 0 & \text{sinon.} \end{cases}$$

La fonction φ est *a priori* quelconque et on peut facilement montrer¹ qu'il est toujours possible de représenter un système (E, φ) à l'aide d'un **graphe de fiabilité** $G = (V, A)$ où

- ▷ $V = \{v_1, v_2, \dots, v_n\}$ est un ensemble de n sommets,
- ▷ A est un ensemble d'arêtes; chaque arête $a \in A$ du graphe correspond à une composante $e \in E$ du système (notons toutefois que plusieurs arêtes du graphe peuvent correspondre à la même composante),
- ▷ le système fonctionne si et seulement s'il existe dans le graphe G une chaîne constituée d'arêtes correspondant à des composantes qui fonctionnent et qui permette de relier les deux sommets particuliers v_1 et v_n .

Exemple (peu réaliste, mais petit) : soit un système comprenant un ordinateur doté de trois ports USB, une souris USB et une imprimante USB. Pour que le système puisse fonctionner (*i.e.* pour que l'on puisse cliquer pour imprimer un document), il faut donc que l'ordinateur, la souris, l'imprimante et au moins deux des trois ports USB fonctionnent. Le graphe de fiabilité représenté sur la figure 1 est une manière d'exprimer cette situation.

Partant au temps $t = 0$ d'un système en état de fonctionner (on suppose qu'initialement toutes les composantes fonctionnent), on appelle **espérance de vie** la durée résiduelle de fonctionnement du système, c'est-à-dire le temps qui va s'écouler jusqu'à ce qu'il tombe en panne. Dans le cadre de ce laboratoire, comme dans la quasi-totalité des applications, le modèle sous-jacent est **stochastique** et l'espérance de vie d'un système est définie comme l'**espérance d'une variable aléatoire** représentant la **durée de fonctionnement** du système jusqu'à ce qu'il tombe en panne. Plus précisément, on supposera que la durée de fonctionnement de la composante $e_i \in E$ est distribuée selon une loi triangulaire $\mathcal{T}(\alpha_i, \gamma_i, \beta_i)$, où α_i, γ_i et β_i sont des paramètres donnés.

¹Par mise en parallèle de l'ensemble des connecteurs minimaux ou par mise en série de toutes les coupes minimales...

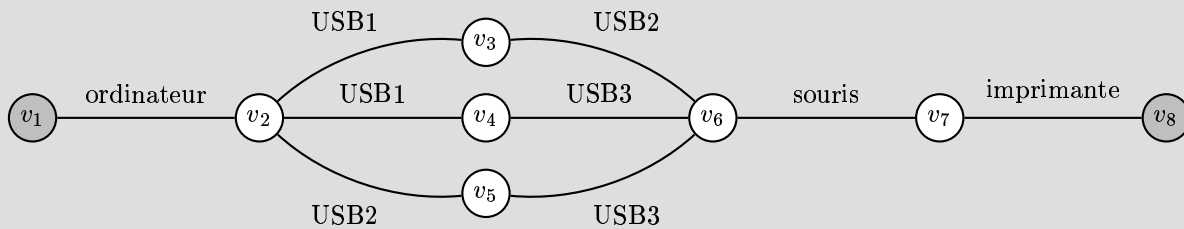


FIG. 1 – Graphe de fiabilité associé au système d'impression décrit ci-dessus.

Dans l'exemple de la figure 1, si l'on admet que les $k = 6$ composantes sont numérotées comme suit, les paramètres (en années) des lois triangulaires associées pourraient être ceux qui sont donnés dans le tableau suivant :

Composante	Description	α	γ	β
e_1	ordinateur	0.2	2.0	3.0
e_2	port USB1	0.5	2.0	3.5
e_3	port USB2	0.5	2.0	3.5
e_4	port USB3	0.5	2.0	3.5
e_5	souris	0.8	3.2	5.0
e_6	imprimante	0.4	2.5	4.0

Le programme qui vous est généreusement fourni se charge de lire la description d'un système à partir d'un **fichier**. Il est par exemple possible de décrire le système de la figure 1 comme suit (fichier data1) :

```

6 8
0.2 2.0 3.0
0.5 2.0 3.5
0.5 2.0 3.5
0.5 2.0 3.5
0.8 3.2 5.0
0.4 2.5 4.0
0
1 0
0 2 0
0 2 0 0
0 3 0 0 0
0 0 3 4 4 0
0 0 0 0 0 5 0
0 0 0 0 0 0 6 0

```

Plus précisément, ce fichier comprend $k + n + 1$ lignes formatées comme suit (voir schéma figure 2) :

- ▷ une ligne spécifiant le nombre de composantes k et le nombre de sommets n ,
- ▷ k lignes précisant les trois paramètres α_i , γ_i et β_i de la loi triangulaire associée à chaque composante,
- ▷ n lignes formant une matrice triangulaire inférieure $(a_{i,j})$ avec $1 \leq i \leq j \leq n$, où

$$a_{i,j} = \begin{cases} 0 & \text{si l'n'y a pas d'arête entre les sommets } v_i \text{ et } v_j, \\ c \in \{1, \dots, k\} & \text{si les sommets } v_i \text{ et } v_j \text{ sont reliés par une arête correspondant à la composante } e_c. \end{cases}$$

Nous n'allons cependant pas nous borner à simplement mesurer l'espérance de vie d'un système ; nous allons également considérer l'impact de **politiques de maintenance** sur cette espérance de vie. Un réparateur est en effet disponible pour **remettre en état** des composantes tombées en panne. On note d la durée d'une réparation (cette valeur est une **constante** positive, identique pour chaque composante). Lorsqu'une réparation est terminée, la composante e_i en question fonctionne à nouveau pour un temps distribué selon une loi $\mathcal{T}(\alpha_i, \gamma_i, \beta_i)$. Lorsqu'une réparation a été commencée, elle est effectuée entièrement (la composante reste hors d'usage jusqu'à la fin de la réparation). Le réparateur ne peut réparer qu'une seule composante à la fois et ne peut pas effectuer de maintenance préventive (*i.e.* « réparer » une composante qui n'est pas en panne).

k	n					
α_1	γ_1	β_1				
α_2	γ_2	β_2				
\vdots	\vdots	\vdots				
α_k	γ_k	β_k				
$a_{1,1}$						
$a_{2,1}$	$a_{2,2}$					
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$				
\vdots	\vdots	\vdots	\ddots			
$a_{n-2,1}$	$a_{n-2,2}$	$a_{n-2,3}$	\cdots	$a_{n-2,n-2}$		
$a_{n-1,1}$	$a_{n-1,2}$	$a_{n-1,3}$	\cdots	$a_{n-1,n-2}$	$a_{n-1,n-1}$	
$a_{n,1}$	$a_{n,2}$	$a_{n,3}$	\cdots	$a_{n,n-2}$	$a_{n,n-1}$	$a_{n,n}$

FIG. 2 – Format de fichier utilisé pour décrire un système.

Trois politiques sont à considérer :

- ▷ politique 1 : ne rien faire (aucune réparation n'est effectuée... on attend que le système tombe en panne),
- ▷ politique 2 : réparer les composantes en panne l'une après l'autre, dans l'ordre d'occurrence des pannes (PPPR : Première en Panne, Première Réparée),
- ▷ politique 3 : politique plus futée à développer par vous-même (vous pouvez notamment exploiter la structure du graphe de fiabilité, les valeurs des paramètres $(\alpha_i, \gamma_i, \beta_i)$, mais aussi l'historique des dates des pannes et des réparations de composantes qui ont eu lieu dans le passé)². Soulignons le fait que cette politique doit être efficace (plus que la politique 2) pour tout jeu de données, pas seulement pour les problèmes spécifiés dans les fichiers `data1`, `data2` et `data3` fournis en exemple.

Code fourni :

L'archive qui vous est fournie sur la page web du cours comprend environ 1300 lignes de code. L'architecture globale du programme, ainsi que la plupart des modules fastidieux (lecture du fichier de données, instanciation du système, gestion de l'échéancier,...) y sont définis. Cette archive comprend 17 fichiers :

- ▷ `MT19937.c` et `MT19937.h` : le générateur pseudo-aléatoire,
- ▷ `generateurs.c` et `generateurs.h` : la bibliothèque de générateurs de variables aléatoires,
- ▷ `echeancier.c` : implémentation de l'échéancier,
- ▷ `echeancier.h` : fichier d'en-tête correspondant et spécification des événements de la simulation,
- ▷ `reseau.c` et `reseau.h` : instanciation du système à partir d'un fichier de données, gestion du fonctionnement des composantes et du système,
- ▷ `simulation.c` et `simulation.h` : génération d'une réalisation de la variable aléatoire « espérance de vie du système » et gestion des événements de la simulation correspondante,
- ▷ `statistiques.c` et `statistiques.h` : analyse statistique des résultats,
- ▷ `Labo1.c` : le programme principal, permettant d'obtenir un intervalle de confiance de seuil à 99% et de demi-largeur inférieure ou égale à une valeur passée en paramètre, pour l'espérance de vie d'un système donné (dans un fichier), lorsqu'une des trois politiques de maintenance est appliquée et pour une durée de maintenance d ,
- ▷ `Makefile` : pour compiler en paix avec la commande `make`,
- ▷ `data1`, `data2` et `data3` : trois fichiers d'exemples de problèmes au format de la figure 2.

Code à réaliser :

Environ 100 lignes de code sont nécessaires pour compléter le programme fourni et effectuer les expériences correspondant aux politiques 1 et 2 (au niveau de votre politique 3, la question du nombre de lignes reste ouverte). Plus précisément, les cinq fichiers suivants sont à rendre (les autres fichiers ne sont pas à rendre et ne **doivent pas être modifiés**) et les éléments suivants sont à réaliser (voir également les commentaires dans le code) :

- ▷ `generateurs.c` : la fonction `generateurs_triangulaire()`,

²Par contre, cette politique n'est pas dotée de dons de voyance et ne peut en aucun cas accéder aux dates des pannes planifiées dans l'échéancier (*i.e.* dans le futur).

- ▷ `reseau.c` : la fonction `reseau_plus_ancienne_comp_en_panne()` pour la politique 2, `reseau_analyse_3()` et `reseau_conseil_3()` pour la politique 3, ainsi que toute fonction/variable auxiliaire que vous jugerez nécessaire d'ajouter (bien que ceci ne soit pas recommandé, notons qu'il est également envisageable de modifier le code fourni dans ce fichier),
- ▷ `simulation.c` : la fonction de gestion des événements `traiter_evenement()`, les fonctions `politique_2()` et `politique_3()`, ainsi que toute fonction/variable auxiliaire que vous jugerez nécessaire d'ajouter (bien que ceci ne soit pas recommandé, notons qu'il est également envisageable de modifier le code fourni dans ce fichier),
- ▷ `statistiques.c` : les quatre fonctions spécifiées,
- ▷ `echeancier.h` : *a priori* il n'y a rien à changer dans ce fichier, sauf si vous tenez à faire quelque chose de spécial (nécessitant l'ajout de nouveaux types d'événements ou de données supplémentaires dans la structure des événements) au niveau de la politique 3.

Autres travaux à effectuer :

1. Lire attentivement cet énoncé, ainsi que le code qui vous est fourni dans l'archive `Labo1.tgz`.
2. Expliciter le schéma de la simulation (notamment la gestion des différents types d'événements).
3. Description/explication/justification de votre politique 3.
4. Pour le problème défini dans le fichier `data1` et lorsque la politique 1 est utilisée, construire des intervalles de confiance (avec seuil de confiance à 99%) pour l'espérance de vie du système, pour des demi-largeurs maximales égales à 0.1, 0.09, 0.08, ..., 0.01 unités de temps. Discuter l'évolution du nombre de réalisations nécessaires en fonction de la précision requise.
5. Pour le problème défini dans le fichier `data1` et lorsque la politique 2 est utilisée, analyser l'évolution de l'espérance de vie du système en fonction de la durée d d'une réparation (fixer la demi-largeur maximale des intervalles de confiance à 0.01 unités de temps). Plus précisément, partir de $d = 1.0$ unité de temps et faire tendre progressivement d vers 0, par exemple jusqu'à $d = 0.0001$ (choisir des valeurs parlantes...).
6. Pour les deux problèmes définis dans les fichiers `data2` et `data3`, lorsque la politique 1 est utilisée, construire un intervalle de confiance à 99% de demi-largeur maximale égale à 0.01 pour l'espérance de vie du système.
7. Pour les deux problèmes définis dans les fichiers `data2` et `data3`, lorsque la politique 2 est utilisée, analyser l'évolution de l'espérance de vie du système en fonction de la durée d d'une réparation (par exemple pour d compris entre 5.0 et 0.1 avec `data2` et pour d compris entre 30.0 et 0.5 avec `data3`). Pour chaque expérience, fixer vous-même la demi-largeur souhaitée des intervalles de confiance en fonction des résultats déjà obtenus. Discuter également ces résultats en les comparant à ceux obtenus aux points 5. et 6.
8. Comparer les performances des politiques 2 et 3 sur la base des problèmes définis dans les fichiers `data2` et `data3`. Il s'agit donc de « vendre »³ votre politique 3. À nouveau, cette analyse est à effectuer en fonction de la durée d des réparations.
9. Remarque : d'une manière générale, lors de la discussion de vos résultats, des représentations graphiques pourraient s'avérer parlantes...
10. Effectuer toute autre analyse que vous jugeriez pertinente.

³À ce propos, notons qu'un concours sera organisé à l'occasion de la correction de vos laboratoires : plus précisément, les performances de votre politique 3 seront confrontées à celles de vos camarades (et/ou professeur) sur la base d'un ou plusieurs autres problèmes que ceux définis dans les fichiers `data2` et `data3`...