UTILISATION DE LA SIMULATION DE MONTE CARLO POUR LA RESOLUTION D'UN BENCHMARK (MINIPLANT)
MONTE CARLO SIMULATION USE FOR RESOLUTION OF A BENCHMARK (MINIPLANT)

Marc BOUISSOU, Hassane CHRAIBI et Sabine MUFFAT
EDF R&D
1 avenue du général de Gaulle 92141 Clamart cedex France

Résumé

Cet article propose de résoudre l'ensemble du cas test MINIPLANT avec les outils présents dans la plate-forme Outils KB3 entièrement développée par EDF. Ce cas test, à vocation pédagogique, représente une installation de production capable de fonctionner avec plusieurs niveaux de production, en fonction des défaillances qui surviennent. Il a l'avantage d'être représentatif de problèmes familiers pour l'analyste : redondances actives et passives, arrangements variés de composants de capacités de production différentes, porte k/n, dépendances liées à la limitation du nombre de réparateurs… L'objectif de l'étude est de quantifier la capacité globale de production. Deux méthodes de résolution seront décrites qui ont la particularité de se baser toutes les deux sur un modèle en langage FIGARO 0, produit automatiquement par KB3 à partir d'une représentation graphique du système. La première méthode consiste à construire et quantifier un arbre d'événements dont les branches terminales correspondent aux différents niveaux de production avec l'outil FIGSEQ dédié à l'exploration de séquences. La seconde méthode est la simulation de Monte Carlo, connue pour fournir des résultats dont la précision est fonction du temps de calcul consenti.

Summary

This article proposes to solve exhaustively the test case MINIPLANT with the tools available in the KB3 workbench, entirely developed by EDF. This pedagogical test case represents a plant able to function with several levels of production, according to the occurrence of failures. It has the advantage of being representative of familiar problems for the analyst: active and standby redundancies, varied arrangements of components of different outputs, k/n redundancies, dependences related to the limitation of the number of repairers... The aim of the study is to quantify the total production capacity. Two resolution methods will be described which have the characteristic to be both based on a model in FIGARO 0 language, automatically produced by KB3 from a graphical representation of the system. The first method consists in building and quantifying an event-tree whose terminal branches correspond to the various levels of production with the tool FIGSEQ dedicated to sequences exploration. The second method is Monte Carlo simulation, known to provide results whose precision is a function of tolerance to high computing times.

Introduction

Depuis plusieurs années, les centrales de production d'électricité possédant des états de fonctionnement intermédiaire cherchent à optimiser leur disponibilité donc leur performance et leurs coûts de maintenance. Ce besoin a conduit l'ISDF à créer en 1998 le cas test MINIPLANT, représentatif de l'ensemble des problèmes rencontrés lors de l'étude d'une telle centrale, pour le confronter aux différents outils existants et comparer différentes approches. L'objectif était de trouver l'outil le plus adapté à la résolution de cette problématique pour ensuite l'exploiter à l'échelle d'une centrale réelle. Dans ce contexte, EDF a proposé deux méthodes originales [1]. Ces méthodes étaient : la construction automatique d'un arbre d'évènements (nous allons la rappeler dans le présent article) et l'utilisation d'un réseau bayésien. Cependant, la version du cas test faisant intervenir des problèmes de dépendances dues au nombre limité de réparateurs n'a pas fait l'objet de résolution par les outils d'EDF de l'époque. D'où l'intérêt aujourd'hui de montrer les capacités de traitement du tout nouvel outil de simulation de Monte Carlo, appelé « YAMS » et développé à EDF.

L'objectif général de l'article est donc d'évaluer de façon précise et détaillée la disponibilité de ce cas test avec l'outil FIGSEQ, basé sur l'exploration de séquences ainsi qu'avec YAMS en montrant leurs capacités de traitement respectives.

Cet article va présenter la résolution des quatre versions du cas test MINIPLANT avec les outils présents dans la plate-forme Outils KB3. Ce vocable désigne un ensemble d'outils développés à EDF, qui capitalisent plus de 15 ans de retour d'expérience sur des études de sûreté de fonctionnement, dans les domaines nucléaire, électrique, télécommunications, automobile… Le cas test est modélisé en langage FIGARO, grâce à l'écriture d'une base de connaissances en langage FIGARO [5][6] qui permet ensuite de saisir graphiquement **tout** système du type de MINIPLANT. Cette base de connaissances modélise des diagrammes de capacité pour l'évaluation de performances de systèmes ayant des états dégradés. Son intérêt est de saisir un schéma fonctionnel, compréhensible par tout non fiabiliste. La base de connaissances étant très simple, elle sera décrite et commentée dans l'article, ce qui permettra de montrer la puissance et la concision du langage FIGARO sur un exemple concret.

La difficulté de résolution du système modélisé vient de la dépendance entre les composants qui impose des outils de résolution de modèles dynamiques. Deux outils de traitement présents dans la plate-forme Outils KB3 permettent de tels traitements mais chacun a aussi ses limites. La première méthode par exploration de séquences avec FIGSEQ est riche d'enseignements par ses résultats quantitatifs et qualitatifs mais selon les modèles, des problèmes d'explosion combinatoire peuvent être rencontrés. Dans ce cas, il est normal d'avoir recours à la simulation de Monte Carlo qui est capable de résoudre tous les types de modèles dynamiques (markoviens et non markoviens). Cependant, les deux inconvénients majeurs de la simulation demeurent l'imprécision des résultats et le temps de calcul élevé. Nous montrerons néanmoins que ces deux approches se complètent et sont faciles à utiliser grâce au formalisme FIGARO et aux outils de la Plate-forme KB3.

Présentation du cas test MINIPLANT

Une première version de l'énoncé de ce cas test a été publiée par EDF dans un compte-rendu du groupe de travail « Recherche Méthodologique » de l'ISDF en 1998. Elle a été modifiée en 1999 pour devenir la version de référence qui a l'avantage d'être plus proche de la réalité industrielle en prenant en compte des dépendances dues à un nombre limité de réparateurs.

Le cas test MINIPLANT (Figure 1) est représenté par un diagramme de fiabilité « élargi » pour permettre de modéliser une installation capable de fonctionner avec plusieurs niveaux de production. Les blocs portent donc une information supplémentaire qui est leur contribution nominale à la capacité de production d'un sous-système exprimée en pourcentage. On peut alors parler d 'un diagramme de **Fiabilité- Capacité**.

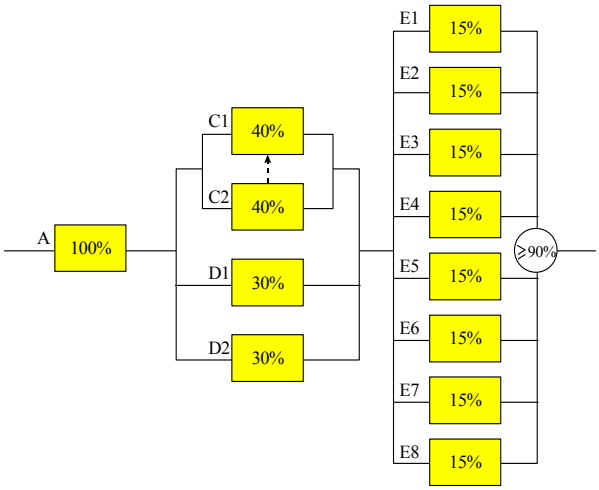


Figure 1 : Structure du système MINIPLANT

Le système se compose de trois sous-systèmes en série. Le premier sous-système (SS1) est représenté par un unique composant A. Le second sous-système (SS2) est un système parallèle ou plutôt deux sous-modules en parallèle. En effet, le premier sous-module est formé de deux composants : C1 initialement actif et C2 initialement en attente car fonctionnant en redondance passive. C2 est donc un composant de secours de C1, qui ne doit fonctionner que suite aux pannes de C1. L'autre sous-module est composé des composants D1 et D2 qui fonctionnent normalement en redondance active. Le troisième sous-système (SS3) composé de huit composants de E1 à E8 n'est opérationnel que si au moins 6 composants parmi ces huit fonctionnent. On obtient alors un seuil de 89% en dessous duquel le sous-système SS3 ne peut plus faire transiter de flux ou de matière.
Tous les composants ont un comportement binaire avec deux états distincts (marche et panne) sauf C2 qui possède trois états : attente, marche, panne.

Nous avons vu que ce diagramme de Fiabilité-Capacité est particulier dans le sens où il permet d'évaluer la capacité globale de production susceptible de transiter de la gauche vers la droite du système. Cette détermination est fonction des règles suivantes :

- R1 : Pour un composant avec X% de capacité nominale, sa capacité à laisser passer le flux est égale à X% si il est disponible et 0% s'il est défaillant.
- R2 : La capacité d'un sous-système, en configuration parallèle, est égale à la **somme** des capacités de ses composants, qui ne peut dépasser 100%.
- R3 : Un sous-système peut être contraint par une valeur de **seuil** au dessous de laquelle il est rendu indisponible. Le SS3 est concerné par cette contrainte qui le rend disponible dès lors que la somme des capacités est supérieure ou égale à 90%. Par conséquent, sa capacité de transit est nulle en dessous de ce seuil.
- R4 : La capacité globale du système, composé de trois sous-systèmes en série, est égale à la plus **faible** des capacités de ces sous-systèmes.

Les différentes réflexions menées pour aboutir au cas test de référence ont conduit à définir quatre variantes du modèle. Ces variantes sont obtenues en croisant deux politiques de maintenance avec deux jeux de données de fiabilité.

Dans les versions 1 et 2, on suppose qu'il n'y a aucune limitation du nombre de réparateurs : dès qu'un composant tombe en panne, sa réparation commence. Dans les versions 3 et 4, on suppose qu'il n'existe qu'un seul réparateur pour chaque sous-système. Chaque réparateur suit la politique FIFO et donc répare les composants dont il a la charge dans l'ordre où ils sont tombés en panne.

Dans les versions 1 et 3, les données de fiabilité à prendre en compte sont celles du Tableau 1. Dans les versions 2 et 4, les données sont les mêmes pour le MTTF mais les MTTR sont multipliés par 10 dans le but de tester la robustesse d'éventuelles approximations.

Composants	**MTTF** (heures)	**MTTR** (heures)	**Capacité** (%)
A	50 000	200	100
C1,C2	10 000	500	40
D1,D2	1 000	10	30
E1 à E8	5 000	100	15

Tableau 1 : Données de Fiabilité - Capacités

Les durées de fonctionnement avant défaillance et les durées de réparation étant des variables aléatoires continues distribuées exponentiellement, les taux de défaillance et de réparation sont immédiatement déduits par :

$$\lambda = \frac{1}{MTTF} \quad \text{et} \quad \mu = \frac{1}{MTTR}$$

Modélisation du cas test

L'objectif global de l'étude est de résoudre l'ensemble du cas test MINIPLANT avec les outils présents dans la plate-forme Outils KB3. Dans un premier temps, nous allons présenter les grands principes de la plate-forme KB3. Nous exposerons ensuite la base de connaissances qui contient toute la connaissance générique des composants permettant de modéliser toute installation de production de type MINIPLANT.

Plate-forme KB3

KB3 est un logiciel permettant d'automatiser les études de sûreté de fonctionnement à partir d'une base de connaissances adaptée aux problématiques du système. A partir d'une représentation graphique de ce système, KB3 génère un modèle textuel dit en FIGARO 0, transparent pour l'utilisateur, qui peut soit être utilisé pour générer automatiquement un arbre de défaillances soit constituer le fichier d'entrée aux outils de traitement (FIGSEQ et YAMS).

Les études de fiabilité et de disponibilité constituent le cœur des activités du département MRI (Management des Risques Industriels) de la Division Recherche et Développement d'EDF. Les modèles utilisés sont souvent construits à la main, ce qui présente de multiples inconvénients parmi lesquels :

- une cohérence souvent imparfaite entre les hypothèses faites par des analystes différents et
- une faible traçabilité des hypothèses (source potentielle d'erreurs lors des réactualisations des études).

D'où les principaux avantages de KB3 qui sont :

- le gain de temps dans la réalisation des études,
- l'accessibilité de l'outil à des non spécialistes du fait de bases de connaissances dédiées à une application,
- la qualité des études obtenue par une garantie de cohérence entre les différents modèles et la traçabilité des données,
- un unique modèle pour plusieurs traitements.

C'est dans ce contexte que le département MRI a lancé un programme de recherche sur le thème de la construction automatique des modèles pour les études de risques. Ce programme a abouti en 1990 à un outil qui permet de construire automatiquement des arbres de défaillances à partir de connaissances génériques sur des classes de composants et de la description physique (saisie graphiquement) du système à étudier : **KB3**. La dernière version KB3 V3, maintenant disponible pour fonctionner sur tous les environnements PC, permet de pérenniser l'outil.

Pour réaliser l'étude de sûreté de fonctionnement d'un système avec KB3, il faut disposer d'une **base de connaissances** préalablement écrite en langage FIGARO. Ce langage a été développé au département MRI.

L'utilisateur de KB3 utilise donc à la fois une base de connaissances adaptée à son application et le logiciel KB3 lui-même qui est constitué d'une interface conviviale pour la description graphique du système étudié et des buts de l'étude. Cette interface permet la construction de modèles de systèmes industriels complexes.

A partir de cette description graphique dans l'IHM de KB3 et des connaissances décrites dans la base, le logiciel KB3 génère un modèle dit **FIGARO** à l'ordre 0. Ce modèle unique est utilisé :

- soit par le générateur d'arbres de défaillances pour les modèles statiques
- soit par le générateur de séquences pour les modèles comportementaux ou Markoviens
- soit par l'outil de simulation de Monte Carlo pour tous les modèles Markoviens ou non.

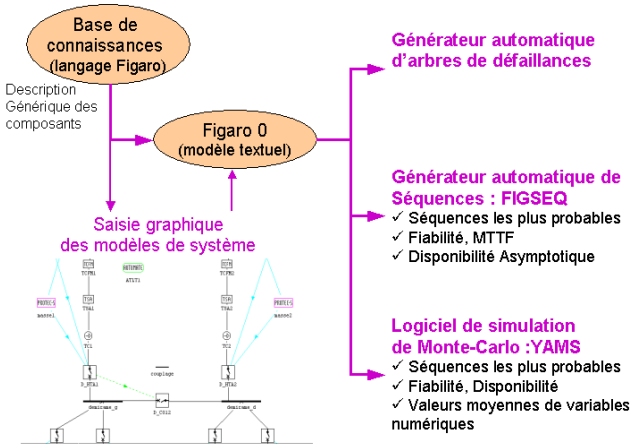


Figure 2 : Principes de la Plate-forme KB3

Outils de traitement de la Plate-forme

Le premier outil de traitement développé par le département MRI est le logiciel **FIGSEQ** pour les modèles comportementaux (Markoviens). Il permet de construire automatiquement les séquences d'événements d'un système à partir d'un modèle KB3 (modèle FIGARO d'ordre 0). FIGSEQ est adapté à l'étude de systèmes comprenant de nombreuses dépendances entre composants (dépendances temporelles notamment) et pour lesquels un modèle structurel de type arbre de défaillances n'est pas adapté. Le principal atout de FIGSEQ tient à ce que la construction du graphe d'états du système n'est pas requise. FIGSEQ permet donc de construire des modèles de séquences sur des systèmes complexes.

Lors d'une analyse par séquences, FIGSEQ produit entre autres les résultats suivants : fiabilité et disponibilité du système, séquences les plus importantes, majorant des erreurs commises…

Le second et récent outil de traitement développé par EDF est **YAMS**, un outil de simulation de Monte Carlo. Son développement permet de proposer une alternative pour le traitement de systèmes Markoviens à grande combinatoire et des systèmes qui présentent plusieurs processus non Markoviens en interaction.

Ce sont ces deux outils qui vont ensuite servir à la résolution du cas test MINIPLANT et feront l'objet d'une comparaison en terme de robustesse des résultats.

Base de connaissances MINIPLANT

Le cas test MINIPLANT est modélisé en langage FIGARO, grâce à l'écriture d'une base de connaissances en langage FIGARO [5][6] qui permet ensuite de saisir graphiquement **tout système du type de MINIPLANT**. Cette base de connaissances modélise des diagrammes de capacité pour l'évaluation de performances de systèmes ayant des états dégradés. Ce modèle peut être traduit en un arbre d'évènements dont les branches terminales correspondent aux différents niveaux de production. Mais l'intérêt est de saisir un schéma fonctionnel, compréhensible par un non fiabiliste.

La base de connaissances étant très simple et concise (90 lignes), elle est décrite et commentée ci-après, ce qui permettra de montrer la puissance et la concision du langage FIGARO sur un exemple concret.

Deux bases de connaissances ont réellement été développées pour permettre la résolution de l'ensemble des versions. Une première base de connaissances a permis de traiter uniquement les deux premières versions en générant un arbre de séquences, étant donné les indépendances entre les composants. Seul le sous-module composé de C1 et C 2 a nécessité de faire appel à un petit graphe de Markov pour résoudre la dépendance entre ces deux composants.
Pour réussir à traiter les versions 3 et 4 avec cette première base, il aurait fallu calculer par une méthode de type graphe de Markov les probabilités de tous les états des trois sous-systèmes en série du diagramme, pour pouvoir affecter ces probabilités aux branches d'un arbre d'évènements. Nous avons préféré axer notre travail sur l'outil de simulation de Monte Carlo et créer une seconde base de connaissances, permettant une automatisation complète de la résolution par la simulation de Monte Carlo.

Nous présentons ci-dessous **l'intégralité de cette seconde base de connaissances**, qui a l'avantage d'illustrer la majorité des constructions syntaxiques du langage FIGARO d'ordre 1.

```
ORDRE_DES_ETAPES
        etape_par_defaut ;
        calcul_rang_max ;
        gestion_rep ;
```

```
TYPE sous_systeme ;
  ATTRIBUT capacite DOMAINE REEL REINITIALISATION 100;
  INTERFACE element GENRE sous_systeme ;
```

```
TYPE sous_systeme_additif SORTE_DE sous_systeme;
  CONSTANTE seuil_fonctionnement
        DOMAINE REEL PAR_DEFAUT  0;
  ATTRIBUT somme_brute
        DOMAINE REEL PAR_DEFAUT  0;
INTERACTION
  SI (SOMME POUR_TOUT x UN element
        DES_TERMES capacite DE x) > seuil_fonctionnement
  ALORS somme_brute <-- (SOMME POUR_TOUT x UN
        element DES_TERMES capacite DE x)
  SINON somme_brute <--  0;

  SI somme_brute >  100 ALORS capacite <--  100
  SINON capacite <-- somme_brute;
```

```
TYPE sous_systeme_min SORTE_DE sous_systeme;
INTERACTION
  SOIT x UN element
  SI capacite(x) < capacite
  ALORS capacite <-- capacite(x) ;
```

```
TYPE composant SORTE_DE sous_systeme;
  INTERFACE   equipe_rep GENRE equipe_reparation
                        CARDINAL 0 JUSQUA 1 ;
  CONSTANTE
     capacite_nominale DOMAINE REEL PAR_DEFAUT 100;
     lambda        DOMAINE REEL PAR_DEFAUT 0.001;
     mu            DOMAINE REEL PAR_DEFAUT 1;
```

```
   PANNE  perte_fonc;
   EFFET   depiler;
   ATTRIBUT
      etat DOMAINE 'arret' 'marche' 'attente_rep' 'en_reparation'
        PAR_DEFAUT 'arret';
      rang DOMAINE ENTIER PAR_DEFAUT 0;

OCCURRENCE
occ_perte_fonc
   SI etat = 'marche'
      IL_PEUT_SE_PRODUIRE
      DEFAILLANCE perte_fonc LOI EXP (lambda)
      PROVOQUE etat <-- 'attente_rep',
             POUR_TOUT e UNE equipe_rep
             FAIRE rang <-- rang_max(e) + 1 ;
occ_reparation
   SI etat = 'en_reparation'
     IL_PEUT_SE_PRODUIRE
     REPARATION rep_perte REPARE  perte_fonc LOI EXP (mu)
     PROVOQUE etat <-- 'marche',
              POUR_TOUT e UNE  equipe_rep
              FAIRE (libre DE e, depiler DE e);

INTERACTION
   SI depiler ET rang > 0
      ALORS depiler <-- FAUX, rang <-- rang - 1;
   SI etat = 'marche'
      ALORS capacite <-- capacite_nominale
      SINON capacite <--  0;

   ETAPE gestion_rep
   SI etat = 'attente_rep' ET (rang = 1 OU CARDINAL(equipe_rep)
      = 0) ET QQSOIT e UNE  equipe_rep ON_A libre DE e
     ALORS etat <-- 'en_reparation',
           POUR_TOUT e UNE  equipe_rep
           FAIRE libre DE e <-- FAUX;
```

```
TYPE bloc_normal SORTE_DE composant;
   ATTRIBUT      etat PAR_DEFAUT 'marche';
```

```
TYPE bloc_secours SORTE_DE composant;
   INTERFACE    secouru GENRE composant;

INTERACTION
declenchement
   SI etat = 'arret' ET QQSOIT x UN  secouru ON_A PANNE DE x
      ALORS etat <-- 'marche';

arret
   SI (etat = 'marche' OU etat = 'sollicite') ET
         IL_EXISTE x UN  secouru TEL_QUE MARCHE DE x
      ALORS etat <-- 'arret';
```

```
TYPE equipe_reparation ;
   INTERFACE cpt_suivi GENRE composant ;
   ATTRIBUT
     libre              DOMAINE BOOLEEN PAR_DEFAUT VRAI ;
     rang_max           DOMAINE ENTIER REINITIALISATION  0 ;
     depiler            DOMAINE BOOLEEN PAR_DEFAUT FAUX ;

INTERACTION
   SI depiler ALORS  depiler <-- FAUX,
      POUR_TOUT c UN  cpt_suivi FAIRE depiler DE c;

   ETAPE calcul_rang_max
   SOIT x UN cpt_suivi
    SI rang(x)> rang_max
      ALORS rang_max <-- rang(x) ;
```

Cette base de connaissances contient la description de 5 types principaux et de deux autres plus amont qui exploitent les avantages de la hiérarchie entre types. Ainsi, aucune information n'est dupliquée dans la base dès lors qu'elle est commune à plusieurs composants.

Pour permettre l'évaluation de la capacité globale de production susceptible de transiter de la gauche vers la droite du système et le respect des 4 règles de calcul, la base de connaissances définit deux opérateurs "sous_systeme_min" et "sous_systeme_plus".

Le "sous-systeme-min" calcule le minimum de la capacité globale entre plusieurs sous-systèmes en série. Grâce au quantificateur SOIT, une seule règle suffit à définir le calcul du minimum des capacités de plusieurs éléments. L'étape d'instanciation, qui permet de faire passer du schéma saisi au fichier FIGARO 0, fichier d'entrée des outils, instancie cette règle générique pour chacun des objets déclarés dans l'interface "element" du sous-système.
Le "sous-systeme-plus" calcule la somme de la capacité globale entre plusieurs sous-systèmes en parallèle (en la limitant à 100). Cet opérateur gère en plus la notion de seuil en dessous duquel la capacité de production devient nulle. La règle qui contient la condition « SI SOMME POUR_TOUT x UN element DES_TERMES capacite DE x) > seuil_fonctionnement » a permis de résumer beaucoup de combinaisons. Le langage FIGARO a une grande faculté de concision grâce à l'utilisation des quantificateurs [5] [6].

Les types "bloc_normal" et "bloc_secours" sont des sous-types de "composant" et ainsi héritent de toute la connaissance déjà décrite dans ce type père. Seuls ces deux types terminaux seront accessibles lors de la modélisation sous KB3. Le type "composant" est le seul à contenir des règles d'occurrence, qui définissent les événements aléatoires (défaillances, fins de réparations) pouvant survenir dans le système. Les autres types ne contiennent que des règles d'interaction décrivant les effets immédiats et certains sur l'ensemble des variables du système, chaque fois qu'un événement aléatoire a lieu.

Le seul mécanisme un peu complexe est celui de la gestion de la pile FIFO des composants à réparer, car il fait intervenir la notion d'ETAPE dans les règles d'interaction ; de plus, le fait de pouvoir déclarer 0 ou 1 équipe de réparation dans l'interface "equipe_rep" de "composant" a imposé l'emploi de quantificateurs. L'avantage est que dans le cas où aucune équipe de réparation n'est déclarée dans cette interface, tout se passe comme si le composant avait sa propre équipe.

Si en revanche, une équipe de réparation doit s'occuper de plusieurs composants, voici ce qui se passe.
Les règles d'occurrence indiquent qu'à la défaillance d'un composant celui-ci prend un rang dans la file d'attente qui est : 1 + le plus grand des rangs des composants en panne (mémorisé dans l'équipe de réparation). D'autre part, un composant en réparation va, à la fin de la réparation, passer dans l'état "marche", en libérant l'équipe de réparateurs et en positionnant à VRAI l'attribut "depiler" de cette équipe.

Lors du déroulement des règles d'interaction qui suit tout événement décrit dans une règle d'occurrence, on a trois étapes. Dans la première, "etape_par_defaut", à laquelle sont rattachées toutes les règles ne faisant pas explicitement mention d'une étape, on calcule les capacités de tous les éléments, on arrête ou démarre les composants de secours en fonction de l'état des composants qu'ils doivent éventuellement remplacer, et on retranche éventuellement 1 au rang de chacun des composants suivis par une équipe de réparateurs. Cette dernière opération a lieu si l'attribut "depiler" de l'équipe de réparateurs est à VRAI ce qui déclenche une règle de "equipe_rep" mettant à VRAI l'effet "depiler" (un effet est une variable booléenne réinitialisée à FAUX à chaque début d'application des règles d'interaction) de tous les composants du groupe (déclarés dans son interface "cpt_suivi"). Ensuite, dans l'étape "calcul_rang_max" le rang maximum des composants en panne est recalculé, par une règle déclarée dans le type "equipe_reparation". Enfin, dans l'étape "gestion_rep" un composant passe en réparation soit si son interface "equipe_rep" est vide, soit si son rang est 1 et l'équipe de réparation est libre.

Modélisation du système MINIPLANT

La première base de connaissances est orientée pour permettre un traitement par FIGSEQ mais uniquement des deux premières versions étant donné les fortes dépendances du système dans les deux autres versions avec un nombre de réparateurs limité et la politique FIFO. Le modèle construit sous KB3 est cependant unique et contient une seule variante pour caractériser les

changements de valeurs de MTTR entre les versions 1 et 2. Dans KB3, les variantes servent au sein d'un même modèle à faire des études de sensibilité.

La seconde base de connaissances (celle décrite plus haut) a l'avantage de permettre le traitement des quatre versions avec l'outil de simulation de Monte Carlo. Nous allons décrire un peu plus précisément ces deux modèles.

Modèle pour FIGSEQ

Pour les versions 1 et 2 résolues par séquences (FIGSEQ), nous avons été amenés à représenter le sous-système {C1, C2}, modélisant une dépendance temporelle, par un composant unique à deux états correspondants à des capacités de 0 et 40%, dont nous avons calculé les probabilités grâce au petit graphe de Markov de la Figure 3.

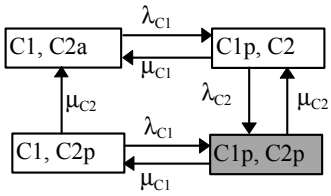


Figure 3 : Graphe de Markov pour {C1,C2}

La probabilité asymptotique de l'état (C1p,C2p) vaut :

- 1.1890606.e-3 dans la variante 1 et
- 7.6923077.e-2 dans la variante 2 du cas test.

La modélisation sous KB3 V3 (**Figure 4**) contient un seul modèle avec la définition d'une seule variante sur la valeur des MTTR et des indisponibilités asymptotiques du composant C1C2 calculé par graphe de Markov.

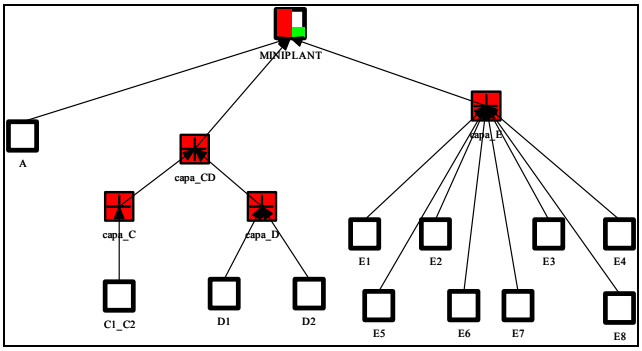


Figure 4 : Modèle des versions 1 et 2 pour FIGSEQ sous KB3V3

Modèle pour Yams

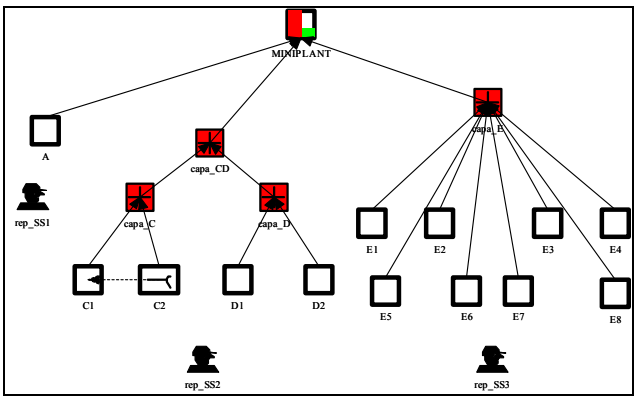
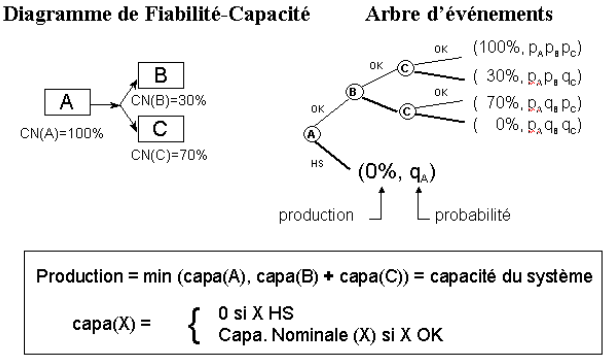


Figure 5 : Modèle des versions 3 et 4 pour YAMS sous KB3V3

Ne pouvant pas exprimer sous la forme d'une variante le fait de disposer d'autant de réparateurs que de composants, nous avons du construire deux modèles différents, le premier pour quantifier les versions 1 et 2 et le second modèle pour les versions 3 et 4 (Figure 5). La seule différence est l'absence des réparateurs dans le premier modèle pour traduire le fait que le système dispose d'autant de réparateurs que nécessaires.

Résolution du cas test par FIGSEQ

La résolution du cas test par FIGSEQ suppose l'indépendance des composants ou au moins de sous-systèmes. Nous allons expliquer le principe à partir du petit exemple suivant :



Dans ce schéma, le flux (d'énergie, de matière, etc..) fonction du type d'installation traverse tout entier le composant A, puis est scindé en deux pour être traité en parallèle par les composants B et C. CN(X) représente la capacité nominale du composant X. En fait la capacité de X à un instant donné est donnée par la dernière expression. La capacité globale de production du système à un instant donné est égale à l'équation « Production ».
Pour faire l'inventaire des divers états possibles du système, avec leurs probabilités et les productions correspondantes, il suffit de construire l'arbre d'évènements. Les formules de calcul des probabilités de branches font intervenir qi, la probabilité de défaillance du composant i (à l'instant t) et pi = 1 – qi. Ces formules exploitent l'indépendance des composants pour effectuer de simples produits. La généralisation à un nombre quelconque de composants et d'états par composants est immédiate.

Cette méthode est évidemment très combinatoire si elle est appliquée telle quelle. Mais deux améliorations permettent d'envisager malgré tout de l'utiliser pour des systèmes complexes.

- L'ordre dans lequel on va choisir d'examiner les composants est important. Si on choisit bien cet ordre (comme c'est le cas ici grâce à un numéro de priorité associé à chaque composant dans la base de connaissances), on peut éviter de développer complètement certaines branches de l'arbre des séquences. En effet, il est souvent facile de prévoir que la production d'un sous-système ou du système complet va prendre l'une de ses valeurs extrêmes (0 ou sa capacité nominale) sans avoir à examiner l'état de tous ses composants. La possibilité de ne pas développer certaines branches de l'arbre d'évènements jusqu'au bout peut permettre de diminuer considérablement la complexité du calcul et ceci sans aucune perte de précision. C'est ainsi que le nombre de branches à développer passe de 2^N à N+1 pour un sous-système composé de N composants en série.
- Il est aussi possible de se limiter à l'ensemble des états de probabilité supérieure à un **seuil**. On perd alors un peu en précision, puisqu'on est amené par conservatisme à considérer que tous les états non explorés (dont on calcule aisément la probabilité) correspondent à une capacité de production nulle.

L'automatisation de la méthode repose sur l'utilisation de l'outil FIGSEQ de recherche automatique de séquences à partir d' un modèle en FIGARO 0. Il suffit de préciser un seuil de probabilité convenable pour limiter l'explosion combinatoire, pour obtenir la liste des combinaisons d'états de composants qui conduisent à divers niveaux de production, avec les probabilités associées.

Pour un système complexe, on ne peut prévoir facilement la liste de toutes les valeurs que peut prendre la production totale. Il faut donc procéder en deux étapes :

- Définition d'états cibles correspondants par exemple à des intervalles de production du type : 0%,]0%, 10%],]10%, 20%], …,]90%, 100%]
- Raffinement de ces intervalles en fonction des productions auxquelles conduisent les séquences prépondérantes en probabilité.

Les résultats du cas test MINIPLANT dans ses versions 1 et 2 ont déjà été publiés [1] avec différents compromis entre temps de calcul et précision mais nous récapitulons dans le tableau en annexe 1 les résultats les plus précis.

Pour la version 1, la quantification complète conduit à une capacité de production espérée en régime stationnaire correspondant à la disponibilité équivalente EA égale à 0.9883, soit à une perte de production moyenne de 1.17%. Par contre, la version 2 connaît une perte de production moyenne de 25.82%. ce résultat s'explique par la très médiocre maintenance intrinsèque des composants qui ont vu leur MTTR multiplié par 10.

Résolution du cas test par YAMS

La résolution du cas test par YAMS, l'outil de simulation de Monte Carlo, n'impose pas l'indépendance des composants. Ainsi, nous allons présenter la quantification des quatre versions.

Le principe de la simulation de Monte Carlo est de simuler un certain nombre d'histoires de durée T. Pour chaque histoire on observe l'état du système aux différentes dates (<T) pour lesquelles on veut calculer des indicateurs de performance.

L'originalité de notre outil de simulation est le fait qu'il permet d'utiliser sans aucune restriction la puissance de modélisation du langage FIGARO, et de définir de manière conviviale des **indicateurs** de performances très variés sans qu'il soit nécessaire de surcharger le modèle. C'est ainsi que l'on peut par exemple :

- calculer l'intégrale d'une variable sur une période de temps,
- calculer le temps de séjour dans une catégorie d'états,
- détecter la première entrée dans une catégorie d'états après un instant donné ou après l'entrée dans une autre catégorie d'états…

Par ailleurs, afin d'optimiser les temps de réponse, il est possible de répartir les calculs sur un ensemble de machines.

La résolution des quatre versions consiste à évaluer :

- la fiabilité au bout d'un an notée R
- l'indisponibilité asymptotique notée U
- les probabilités asymptotiques associées aux différents niveaux de production possibles
- la disponibilité équivalente notée EA [1] ou la perte de capacité de production moyenne.

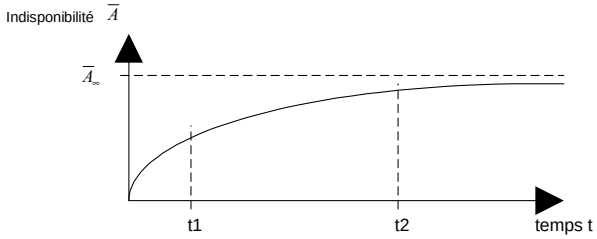
La description des traitements dans YAMS repose sur deux types de données :

1. Données principales de **paramétrisation** des traitements :
   - Temps de mission et instants de mémorisation des histoires. Cette donnée définit l'intervalle de temps sur lequel se feront les simulations ainsi que les instants pour lesquels certains indicateurs seront calculés pour être mémorisés.
   - Nombre maximum d'histoires simulées.
2. Données de définition des **résultats** attendus :
   - **Liste des états**. Un état est défini comme une expression booléenne, de type FIGARO, qui met en œuvre les variables du système. Un état pourra être qualifié de **cible**. Dans ce cas, YAMS le considère comme absorbant, même s'il ne l'est pas d'après le comportement du modèle. Sinon, l'état pourra être qualifié d'état **d'analyse**. Ces deux catégories d'états sont adaptées respectivement à des calculs de fiabilité (en optimisant le temps de calcul) et de disponibilité.
   - **Liste d'indicateurs**. Chaque indicateur est le résultat d'une **fonction** (par exemple TEMPS_DE_SEJOUR, DEJA_REALISE, INTEGRALE …) appliquée à une expression FIGARO quelconque.

Pour la résolution de MINIPLANT, nous avons utilisé :

- La fonction DEJA_REALISE pour le calcul de fiabilité,
- La fonction indicatrice d'un état ou la fonction TEMPS_DE_SEJOUR pour le calcul de l'indisponibilité.

Nous avons utilisé deux fonctions différentes pour le calcul de l'indisponibilité asymptotique afin d'optimiser la précision des résultats pour un temps de calcul donné.
En effet, pour les probabilités fortes, il vaut mieux utiliser la fonction indicatrice, alors que pour les probabilités faibles, c'est avec la fonction TEMPS_DE_SEJOUR qu'on a le meilleur résultat. Cette constatation est facile à expliquer à partir de la courbe ci-dessous, qui représente l'évolution de l'indisponibilité en fonction du temps.



$$\overline{A}_{moy}(T) = \frac{1}{T} \times \int_0^T \overline{A}(t)dt$$

$$\lim \overline{A}_{moy}(T) \xrightarrow[T \to \infty]{} \overline{A}_\infty$$

On constate expérimentalement que ∀ t, l'intervalle de confiance est meilleur pour $\overline{A}_{moy}(t)$ (calculé grâce à la fonction TEMPS_DE_SEJOUR) que pour $\overline{A}(t)$ (calculé grâce à la fonction indicatrice). Mais si la valeur asymptotique $\overline{A}_\infty$ est "grande", la convergence de $\overline{A}_{moy}$ vers $\overline{A}_\infty$ est lente et il faut prendre un temps de mission très grand pour chaque histoire, ce qui augmente le temps de calcul.

Il serait intéressant de voir s'il est possible de déterminer la meilleure méthode d'estimation en fonction de la probabilité à calculer pour un effort de calcul fixé.

L'exploitation des résultats (Annexe : Tableau 3 et Tableau 4) conduit à une disponibilité équivalente égale à 0.9756, soit 2.44% de perte moyenne pour la version 3 et à 0.217, soit 78.23% de perte moyenne pour la version 4. Ce dernier résultat pour la version 4 est « irréaliste » et s'explique par la faible disponibilité de l'unique réparateur alloué à chaque sous-système.

COMPARAISON DES METHODES DE RESOLUTION

La résolution du cas test MINIPLANT avec deux méthodes de résolution a permis de se rendre compte des avantages et des limites des outils.
Il est important de remarquer qu'une fois les bases de connaissances disponibles, il est aisé de mettre en œuvre les deux techniques à partir d'un modèle unique saisi graphiquement avec l'interface de KB3. Il est donc possible à coût très faible de profiter de tous les outils présents dans la Plate-forme KB3. La modélisation du cas test MINIPLANT avec KB3 a l'avantage d'être très proche du comportement physique des systèmes, d'où une accessibilité assurée pour des non fiabilistes.

Les principaux avantages de FIGSEQ sont de fournir des résultats qualitatifs (combinaisons de défaillance conduisant aux divers niveaux de production) en plus des résultats quantitatifs et de permettre d'étudier des systèmes de grande taille, grâce à la possibilité de faire un traitement approximatif. En revanche, la technique d'arbre d'événements ne permet que des calculs de disponibilité et n'est pas applicable lorsque de fortes dépendances existent entre les composants, comme dans les versions 3 et 4 de MINIPLANT.

Pour pallier les restrictions de FIGSEQ, nous avons pu utiliser un outil de simulation de Monte Carlo qui présente l'avantage de résoudre tous les cas, même à l'échelle d'un système réel, avec une seule limite qui peut être le temps de traitement.

CONCLUSION

Dans cet article, nous avons non seulement donné les résultats complets du cas-test MINIPLANT, mais aussi montré comment, à l'aide d'un développement de très faible coût (une base de connaissances FIGARO d'une centaine de lignes) il était possible d'automatiser complètement ce type de calculs pour **tout système présentant des caractéristiques semblables à celles du cas-test**, à partir d'une représentation graphique très simple et naturelle.

Les limitations propres à chacune des méthodes déjà exploitées pour résoudre MINIPLANT (cf. [1], [2], [3], [4]) confirment que la simulation de Monte-Carlo reste, en dépit de sa lourdeur calculatoire, une méthode incontournable car elle répond totalement aux exigences d'évaluation de performances **et** elle est capable de passer à l'échelle réelle.

En effet, la démonstration effectuée sur un exemple pédagogique est facilement transposable à l'échelle réelle, au prix d'une augmentation du temps de calcul à peu près proportionnelle à la taille du système à traiter ; c'est ce qui fait toute la richesse et la robustesse des outils de la Plate-forme KB3.

REFERENCES

[1]. M. Bouissou
Deux méthodes originales pour calculer les performances d'un système possédant des états de fonctionnement dégradé. LambdaMu 12, 12ème colloque de fiabilité et maintenabilité, Montpellier (France), Mars 2000.

[2]. J.P. Signoret
Cas-test MINIPLANT – calculs approchés, compte-rendu de la réunion du groupe « Recherche Méthodologique » de l'ISDF, 06/05/1999.

[3]. J.L.Chabot, Y. Dutuit, A. Rauzy, J.P. Signoret
MINIPLANT, un cas pédagogique de sûreté de fonctionnement, LambdaMu 12, 12ème colloque de fiabilité et maintenabilité, Montpellier (France), Mars 2000.

[4]. M. Bouissou, O. Pourret
A Bayesian belief network based method for performance evaluation and troubleshooting of multi-state systems, International Journal of Reliability, Quality and Safety Engineering, Vol.10, No. 4 (December 2003) p.407-416.

[5]. M. Bouissou, S. Humbert, S. Muffat, N. Villatte
KB3 Tool: Feedback On Knowledge Bases, ESREL 2002, Lyon (France), Mars 2002.

[6]. M. Bouissou, J.C. Houdebine
Inconsistency Detection In KB3 Models, ESREL 2002, Lyon (France), Mars 2002.

ANNEXE

Tableau récapitulatif des résultats pour le cas test MINIPLANT										
Outil	Versions	Probabilités asymptotiques des capacités de production pour l'installation complète							EA	Commentaires
		0%	30%	40%	60%	70%	90%	100%		
FIGSEQ	V1	0.00437	0.0000232	0.0000974	0.00116	0.01949	0.00932	0.9655	0.9883	Temps calcul : <5''
	V2	0.16865092	0.01057	0.006346	0.05288	0.1269	0.19106	0.44355	0.742	Temps calcul : <5''

Tableau 2 : Probabilités associées à des pourcentages de production pour les versions 1 et 2 du cas test MINIPLANT - résultats obtenus avec FIGSEQ

Tableau récapitulatif des résultats pour le cas test MINIPLANT										
Outil	Versions	SS1		SS2		SS3		Système Complet		Commentaires
		R(8760h)	U	R(8760h)	U	R(8760h)	U	R(8760h)	U	
YAMS	V1	0.8376	0.00398	0.9996	1.387e-7	0.9056	0.000394	0.7574	0.004374	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 5'
	V2	0.8392	0.03735	0.9646	0.00058	0.1226	0.13237	0.102	0.16516	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 4'20''
	V3	0.8338	0.003948	0.9864	0.000759	0.8484	0.002466	0.6964	0.007161	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 5'
	V4	0.8404	0.037296	0.8688	0.122499	0.0606	0.635733	0.0448	0.689468	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 4'

Tableau 3 : Fiabilité (R) et Indisponibilité (U) pour les quatre versions du cas test MINIPLANT - résultats obtenus avec YAMS

Tableau récapitulatif des résultats pour le cas test MINIPLANT									
Outil	Versions	Probabilités des capacités de production pour l'installation complète							EA	Commentaires
		0%	30%	40%	60%	70%	90%	100%		
YAMS	V1	0.004374	2.343e-5	9.683e-5	0.001184	0.019518	0.0096	0.9648	0.9879	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 5'
	V2	0.165	0.0116	0.006389	0.0556	0.1218	0.1974.	0.441	0.7433	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 4'20''
	V3	0.007161	0.000979	0.007966	0.000561	0.0368	0.0218	0.9264	0.9756	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul :5'
	V4	0.7148	0.002556	0.0698	0.000573	0.0556	0.0748	0.0824	0.2177	Nbre hist. : 5000 Temps mission : 10^5 Temps calcul : 4'

Tableau 4 : Probabilités associées à des pourcentages de production pour les quatre versions du cas test MINIPLANT - résultats obtenus avec YAMS