

État de l'art

Le projet se concentre sur le problème de *Troubleshooting*, c'est-à-dire sur le problème suivant : étant donné un dispositif en panne, on cherche une stratégie de réparation de coût total minimal. Plus précisément, on considère que le dispositif est constitué d'un nombre fini de composantes c_1, \dots, c_n , certaines pouvant être en panne, et que l'on dispose de 2 types d'actions qui peuvent être réalisées de façon séquentielle : observations et réparations.

Les observations, que l'on dénote par o_1, \dots, o_m , peuvent être "locales", c'est-à-dire d'une seule composante du dispositif, ou "globales" quand elles dépendent de plusieurs composantes. Il peut y avoir de composantes qui n'ont pas d'observation locale associée. On considère aussi qu'on a une observation spéciale o_0 qui porte sur l'état général du dispositif. Les réparations portent toujours sur une seule composante à la fois et on note r_i la réparation de la composante i .

Les ensembles d'observations, réparations et actions sont dénotés respectivement par $\mathcal{O} = \{o_0, \dots, o_m\}$, $\mathcal{R} = \{r_1, \dots, r_n\}$ et $\mathcal{A} = \mathcal{O} \cup \mathcal{R}$. Chaque action $a \in \mathcal{A}$ a un coût associé $C(a) \geq 0$ et l'objectif est donc de mettre le dispositif en état de marche en minimisant le coût total $\sum_i C(a_i)$, où a_i appartiennent à l'ensemble des actions prises. Dans certaines situations, il peut être intéressant de rajouter à \mathcal{A} une action spéciale, a_0 , correspondant à "appeler le service", qui résout le problème avec certitude mais a un coût très élevé, représentant, par exemple, la possibilité d'envoyer le dispositif à un centre plus spécialisé ou d'en acheter un nouveau.

Le problème de minimisation présenté comporte des difficultés liées aux incertitudes: on ne connaît pas quelles sont les composantes défectueuses, quels seront les résultats des observations ni les conséquences d'une réparation sur l'ensemble du dispositif. Il nous faut ainsi prendre des décisions dans l'incertain et, afin de traiter ce problème, on se sert des outils de la théorie de la décision telle que décrite de façon générale dans [7]. Il s'agit d'un cadre formel qui permet de faire des choix d'actions parmi des alternatives lorsque les conséquences de ces actions ne sont connues que dans un sens probabiliste. Cette théorie repose sur la modélisation probabiliste et la représentation des préférences d'un agent, de façon synthétique, à travers une fonction d'utilité.

L'idée d'une fonction d'utilité est de donner des valeurs numériques à des résultats. Une hypothèse fondamentale faite pour cela est de supposer que chaque paire de résultats peut être comparée : un des résultats sera forcément meilleur ou aussi bon que l'autre. On demande aussi à ce que cette comparaison des résultats soit transitive : préférer A à B et B à C implique préférer A à C . Une autre hypothèse fondamentale est que l'on peut comparer non seulement des résultats purs mais aussi des loteries, c'est-à-dire des situations où l'on peut avoir quelques résultats avec une certaine probabilité. En particulier, si un agent préfère A à B , alors, si on lui donne le choix entre deux loteries entre A et B , l'agent préférera la loterie donnant plus de probabilité à A .

Les loteries n'ont pas de valeur intrinsèque, ce qui implique que l'on n'a pas intérêt à faire des loteries imbriquées, où le prix d'une loterie serait de participer à une deuxième loterie. Finalement, on suppose une continuité des loteries: si l'agent a l'ordre de préférence $A > C > B$, alors il existe une loterie entre A et B telle que l'agent sera indifférent entre cette loterie et le résultat C . Sous ces hypothèses, il est possible de condenser les préférences de l'agent dans une fonction d'utilité u telle que $u(A) > u(B)$ si et seulement si l'agent préfère A à B . En plus, si l'agent est indifférent entre C et une loterie entre A et B avec probabilités respectives P et $1 - P$, alors $u(C) = Pu(A) + (1 - P)u(B)$, c'est-à-dire, l'utilité de la loterie est l'espérance de son gain.

Dans un cadre simple avec une seule action à prendre parmi $\alpha_1, \dots, \alpha_k$ et des résultats possibles entre R_1, \dots, R_ℓ , on calcule l'espérance de l'utilité de l'action α_i par $Eu(\alpha_i) = \sum_{j=1}^{\ell} u(R_j)P(R_j | \alpha_i)$ et on choisit celle qui maximise cette utilité espérée. En général, on doit faire face à des problèmes avec une séquence de décisions que l'on peut représenter par un arbre, comme celui de la Figure 1, mais dont le calcul exhaustif en général est trop complexe, nécessitant ainsi de méthodes d'approximation permettant de résoudre le problème en temps raisonnable.

Dans le cadre du *Troubleshooting*, les actions α_i sont des observations ou réparations de \mathcal{A} . L'utilité d'un résultat R_i est le coût des actions qui mènent de la racine à la feuille R_i , la maximisation de l'utilité est remplacée par la minimisation du coût, et tous les résultats R_i représentent

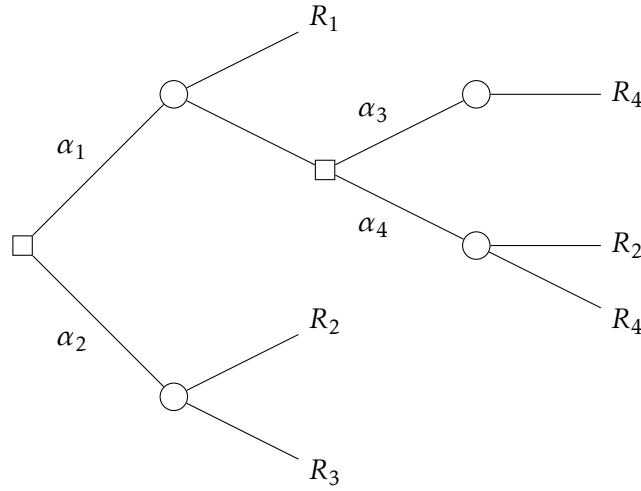


FIGURE 1 – Arbre avec une séquence d’au plus deux décisions à prendre. Les nœuds carrés représentent les décisions à prendre et ceux en forme de cercle, les loteries.

la même situation, à savoir le fonctionnement normal du dispositif.

Les problèmes de *Troubleshooting* ont été efficacement modélisés de façon approchée en utilisant des réseaux bayésiens dans [2, 3]. Une première approche utilise des hypothèses assez restrictives et donne un algorithme final très efficace. Plus précisément, on suppose qu’il n’y a qu’une seule composante présentant un défaut, que les coûts des réparations sont indépendants, et que la seule observation possible est celle de l’état du dispositif, o_0 , qui a un coût 0. On suppose aussi qu’on dispose des probabilités p_i que la réparation r_i résout le problème. Dans ce cas, l’algorithme consiste à calculer les rapports $\frac{p_i}{C(r_i)}$ et réparer les composantes dans l’ordre décroissant de ces rapports, observant après chaque réparation si le dispositif marche ou pas.

Les hypothèses précédentes sont trop restrictives car, d’une part, il n’est pas réaliste de supposer qu’on n’a qu’une seule composante en panne et, d’autre part, on dispose souvent d’autres observations outre o_0 et les informations qu’elles peuvent apporter peuvent être assez importantes pour que l’on les ignore. Cette deuxième remarque est en lien avec la notion de *valeur de l’information* : la valeur qu’une information apporte, dans le cadre du *Troubleshooting*, est la différence entre le coût espéré de réparation sans cette information et le coût en prenant en compte l’information (auquel on ajoute aussi le coût d’obtention de l’information à travers une observation). Cette notion s’applique à des problèmes de décision plus généraux que le *Troubleshooting*, comme décrit dans [1] pour les problèmes d’élicitation décrit plus en détail ci-après.

Ainsi, dans un second temps, [2, 3] fait des hypothèses moins restrictives. On suppose désormais qu’il peut y avoir plusieurs composantes en panne mais on restreint les observations que l’on peut faire. Pour les composantes non-observables (c’est-à-dire, qui n’ont pas d’observation locale associée), la seule action disponible est sa réparation. Pour les composantes observables, on impose de toujours faire l’observation locale correspondante avant la réparation, ce qui est appelé une paire observation-réparation. Ainsi, on restreint l’ensemble \mathcal{A} des actions possibles aux réparations de composantes non-observables et aux paires observation-réparation pour les autres composantes. Le coût de la paire (o_i, r_j) est $C(o_i) + P(o_i \neq \text{normal} \mid E)C(r_j)$, où E représente les informations dont on dispose. L’algorithme suit la même idée que le précédent, en choisissant à chaque étape le plus grand rapport probabilité/coût, mais ces rapports doivent désormais être recalculés à chaque étape car les probabilités et les coûts évoluent en fonction des actions déjà effectuées.

Finalement, [2, 3] rajoute la possibilité de faire des observations globales en dehors des paires observation-réparation. Pour éviter la complexité du cas général, ils développent une technique appelée *myope*, qui consiste à calculer l’espérance de coût après une observation globale o_i de façon approchée en supposant qu’aucune autre observation globale ne sera faite dans la suite. À chaque étape, on compare ces espérances de coût (une pour chaque observation globale) avec l’espérance

de coût sans observation (celle que l'on obtiendrait en appliquant l'algorithme précédent) pour décider s'il est intéressant de réaliser une observation globale à cette étape ou pas. Cela revient aussi à déterminer si la valeur de l'information apportée par l'observation est positive ou pas.

Ces travaux présentent aussi des méthodes pour calculer les probabilités de réparation en utilisant des réseaux bayésiens. Pour ce faire, il est nécessaire non seulement de calculer ces probabilités mais aussi de les mettre à jour en fonction des informations acquises lors d'observations et de réparations. Afin de simplifier ce calcul, les articles introduisent la notion de réseaux de réponse, construits à partir d'un réseau bayésien et d'une action effectuée. Des simplifications supplémentaires sont encore possibles sous l'hypothèse d'indépendance causale. Cet algorithme a été testé et validé pour certains modèles concrets.

Des extensions de l'approche de [2, 3] ont été proposées en particulier dans [4, 5] où les méthodes développées ont permis d'obtenir des résultats assez efficaces pour des cas plus généraux. Plus spécifiquement, l'article [4] considère, d'abord, que des composantes et des actions de réparation peuvent être séparées et que chaque telle action permet de traiter une panne avec une certaine probabilité. Ainsi, les actions ne sont plus parfaites et ne conduisent pas toujours vers une réparation d'une composante. Par ailleurs, l'on suppose que quelle que soit action a une possibilité de dépanner plusieurs composantes et, réciproquement, une composante quelconque peut être réparée par certaines actions différentes. De plus, cet article propose une approche qui améliore une technique myope dont l'on a parlé au-dessus. L'on discutera comment ces modifications influencent un rapport d'approximation du problème ci-dessous. Quant à l'article [5], l'on y suppose un cas encore plus général où chacune composante consiste aux "sous-composantes" qui elles-mêmes produisent une panne d'un appareil et auxquelles l'on peut appliquer une action de réparation. En outre, des composantes comme des ensembles de sous-composantes ne sont pas forcément deux-à-deux disjoints. En conséquence, dans ce modèle il est possible qu'une sous-composante X soit une part de deux composantes différentes, c_i et c_j , $i \neq j$. L'on peut noter que selon des résultats de simulations numériques fournis dans [5] les algorithmes y développés retournent des stratégies de troubleshooting beaucoup plus efficaces qu'une approche myope pour des problèmes concrets posés à l'entrée. En effet, pour les modèles observés une technique de [4, 5] retournait des solutions dont le coût espéré de réparation avait un taux relatif moyen (average relative difference) de 2.51% de l'optimum trouvé par une recherche exhaustive au lieu de 21.5% pour l'approche myope.

Par ailleurs, on remarque aussi qu'il existe des résultats sur la complexité du problème. En effet, l'on peut démontrer que le problème considéré appartient à NP dans le cas général [8]. De plus, une complexité d'approximation a été étudiée dans [6].

Dans les problèmes de la théorie de la décision en général, la fonction d'utilité n'est pas parfaitement connue et il nous faut alors des mécanismes d'élicitation permettant de l'estimer. Cela est bien le cas du problème de *Troubleshooting* qui nous intéresse : malgré le fait que l'utilité provient du coût et que les coûts des actions individuelles sont connus, la connaissance du coût de réparation de façon exacte impliquerait un parcours complet de l'arbre des décisions, ce qui n'est pas réalisable dans la plupart des cas. L'article [1] présente le problème d'élicitation des fonctions d'utilité et donne un panorama des techniques pour le résoudre.

L'idée principale est de considérer que la fonction d'utilité dépend des résultats à travers d'un certain nombre de caractéristiques de ces résultats. Autrement dit, on représente un résultat R comme un vecteur de caractéristiques (x_1, \dots, x_d) et on regarde $u(R)$ comme $u(x_1, \dots, x_d)$. L'article [1] suppose alors que u satisfait une hypothèse d'indépendance additive généralisée, ce qui permet de l'écrire comme combinaison linéaire de fonctions u_1, \dots, u_p , chacune dépendant seulement d'une partie des caractéristiques x_i , par exemple $u(x_1, x_2, x_3, x_4) = \lambda_1 u_1(x_1) + \lambda_2 u_2(x_2, x_4) + \lambda_3 u_3(x_3, x_4)$. Ainsi, l'élicitation peut être décomposée en deux étapes, une locale correspondant à estimer les u_i et une globale afin de déterminer les λ_i . Il présente aussi deux techniques pour représenter les incertitudes sur la fonction d'utilité, basées sur une approche bayésienne et une approche ensembliste. Celle qui nous intéresse est la bayésienne, qui repose sur la notion de valeur de l'information, comme expliqué précédemment.

Pour ce projet, on commence par une réalisation d'un logiciel qui permettra de résoudre des problèmes de *Troubleshooting* différents à partir des leurs modèles donnés sous une forme de réseau bayésien et en utilisant les algorithmes décrits dans les références ci-dessus, notamment [2,3]. On cherchera ensuite des améliorations possibles pour ces algorithmes.

Mots-clés : Troubleshooting, Value of Information.

Références

- [1] D. Braziunas and C. Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4):79, dec 2008.
- [2] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. Technical report, Microsoft Research Technical Report MSR-TR-94-07, 1994.
- [3] D. Heckerman, J. S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, mar 1995.
- [4] F. V. Jensen, U. Kjærulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15(4):321–333, sep 2001.
- [5] H. Langseth. Decision theoretic troubleshooting of coherent systems. *Reliability Engineering & System Safety*, 80(1):49–62, apr 2003.
- [6] V. Lín. Decision-theoretic troubleshooting: Hardness of approximation. *International Journal of Approximate Reasoning*, 55(4):977–988, jun 2014.
- [7] D. North. A tutorial introduction to decision theory. *IEEE Transactions on Systems Science and Cybernetics*, 4(3):200–210, 1968.
- [8] M. Vomlelová. Complexity of decision-theoretic troubleshooting. *International Journal of Intelligent Systems*, 18(2):267–277, jan 2003.