1 État de l'art

1.1 Description générale du problème de *Troubleshooting*

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 $c_1, \ldots, c_n$, 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 $o_1, \ldots, o_m$, 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 $\mathcal{O} = \{o_0, \ldots, o_m\}$, $\mathcal{R} = \{r_1, \ldots, r_n\}$ 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.

1.2 Théorie de la décision et fonctions d'utilité

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 [15]. 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 (axiome de *complétude*). 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 (axiome de *transitivité*). 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 (axiome d'*indépendance*). 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 (axiome de *continuité*). Sous ces hypothèses de complétude, transitivité, indépendance et continuité, d'après le Théorème d'utilité de von Neumann–Morgenstern [19], 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 l'utilité du gain.

Dans un cadre simple avec une seule action à prendre parmi $\alpha_1, \ldots, \alpha_k$ et des résultats possibles entre $R_1, \ldots, R_\ell$, on calcule l'espérance de l'utilité de l'action α_i par

$$Eu(\alpha_i) = \sum_{j=1}^{\ell} u(R_j)P(R_j \mid \alpha_i)$$

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.

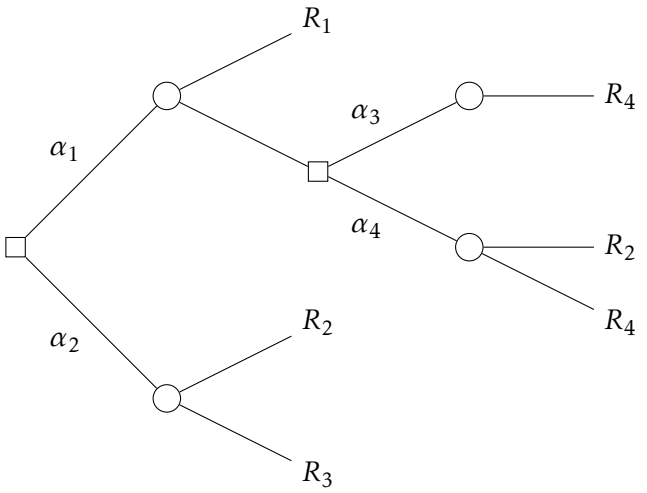

FIGURE 1 – Arbre avec une séquence d'au plus deux décisions à prendre. Les nœuds carrés représentent les décisions à prendre et ceux en forme de cercle, les loteries.

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

1.3 Valeur de l'information myope

Les concepts de bases de la théorie de la valeur de l'information sont présentés dans l'article [9]. Plus précisément, supposons que l'on doit prendre une décision D dans l'incertain parmi celles proposées sachant que l'on connaît un ensemble d'évidences ε. Supposons en plus qu'il existe un voyant qui est capable de fournir une autre évidence X, ce qui diminue les incertitudes. Néanmoins, obtenir X ne serait pas gratuit : pour qu'un voyant nous dise son secret il faut lui payer C_X. Serait-ce alors mieux de payer au voyant ou doit-on prendre plutôt une décision sans informations additionnelles ? L'idée principale de l'approche proposée dans [9] est de comparer l'utilité que l'on obtiendrait sans et avec l'évidence fournie par le voyant. Cette approche est myope car on considère que l'on ne cherchera pas d'autres informations après avoir obtenu X.

Si $Eu(D \mid \varepsilon)$ est l'utilité espérée lorsque l'on prend la décision D sachant ε, selon la technique proposée, il suffit de connaître la valeur

$$\text{EVOI}(X, \varepsilon) = \max_D \sum_{x_i} Eu(D \mid X = x_i, \varepsilon)P(X = x_i \mid \varepsilon) - \max_D Eu(D \mid \varepsilon), \tag{1}$$

donc la différence entre l'utilité espérée maximale si l'on connaît X et celle sans connaître X. Selon ce critère, on décide alors de payer pour évidence X si $Eu_X > 0$ et prendre une décision directement sinon. De plus, si l'on avait un choix entre des évidences $X_1, X_2, \ldots, X_n$, on choisirait celle qui apporterait à Eu_X une valeur maximale toujours en vérifiant que $Eu_X > 0$.

Cette idée est aussi celle utilisée dans l'approche myope décrite dans la Section 1.5.3. Après avoir considéré que les paires observation-réparation sont des actions élémentaires, on peut voir l'introduction des observations globales comme des actions pour obtenir plus d'information. Le choix si une de ces actions sera prise ou pas est fait à travers un critère tout à fait analogue à celui décrit par (1).

1.4 Réseaux Bayésiens

Les incertitudes dans le problème de *Troubleshooting* ont plusieurs origines : on ne connaît pas quelles composantes sont en panne ni les effets précis que le changement de l'état d'une composante peut avoir sur les autres autres composantes, sur les observations et sur l'état du système. Ces incertitudes sont représentées dans notre approche par des probabilités. La représentation intégrale de la loi de probabilité jointe de toutes les composantes et de toutes les observations du système serait trop gourmande en mémoire et inutilement complexe puisque l'on peut imaginer qu'il y a plusieurs relations d'indépendance ou d'indépendance conditionnelle entre elles. Ainsi, les réseaux Bayésiens, décrits par exemple dans [12], sont un outil mathématique adaptée à la représentation des probabilités de notre problème.

Les réseaux Bayésiens permettent de simplifier la représentation des lois de probabilité grâce aux relations d'indépendance conditionnelle entre les variables. Plus précisément, un réseau Bayésien est défini comme un graphe orienté acyclique dans lequel les nœuds représentent les variables d'intérêt et, à chaque nœud X, on associe une loi de probabilité conditionnelle du type

$$P(X \mid Y_1, \ldots, Y_n),$$

où $Y_1, \ldots, Y_n$ sont les parents immédiats de X dans le graphe (ou la loi de probabilité de X si X n'a pas de parents dans le graphe). Dans cette situation, conditionnellement à ses parents immédiats $Y_1, \ldots, Y_n$, un nœud X est indépendant de tout autre nœud qui n'est pas un de ses descendants. La loi de probabilité jointe de toutes les variables peut être déterminée par multiplication des lois de probabilité de chaque nœud.

Pour la construction d'un tel graphe dans des situations pratiques, les flèches ne représentent pas forcément de lien causal, puisque les parents influencent uniquement la *probabilité* des enfants. L'important est que deux nœuds conditionnellement indépendants dans le graphe représentent bien des variables conditionnellement indépendantes dans le cas pratique en question. Pour la manipulation des réseaux Bayésiens, on utilise la bibliothèque `pyAgrum` de Python [6], qui contient, parmi d'autres modules, une implémentation efficace et facilement manipulable de ce type de réseau.

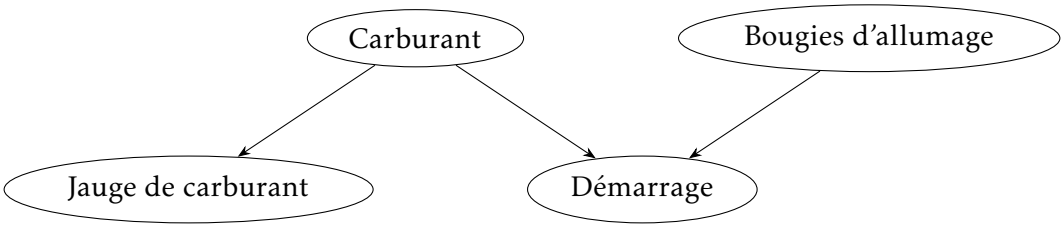

FIGURE 2 – Exemple de réseau bayésien pour le problème de démarrage de la voiture

Un exemple classique de réseau Bayésien pour le *Troubleshooting*, présenté dans [12], est celui du *problème de démarrage de la voiture*, dans lequel on considère une voiture qui ne démarre pas. La Figure 2 représente ce réseau, avec quatre nœuds représentant quatre variables de la voiture (démarrage ou pas, affichage de la jauge de carburant, présence de carburant, fonctionnement

des bougies d'allumage). Dans ce réseau, par exemple, la probabilité que la voiture démarre est exprimée comme une loi conditionnelle dépendant de la présence de carburant et du fonctionnement des bougies. La probabilité que la jauge de carburant affiche qu'il y a du carburant ou pas est représentée comme une loi conditionnelle dépendant uniquement de la présence de carburant. Le fonctionnement des bougies et la présence de carburant sont représentés par deux lois de probabilité simples, sans conditionnement.

1.5 Approches au problème de *Troubleshooting*

1.5.1 Approche exacte

L'approche immédiate pour résoudre le problème *Troubleshooting* est de construire l'arbre correspondant avec toutes les actions d'observation et réparation qui peuvent être réalisées à chaque étape, les feuilles correspondant aux états où le dispositif a été réparé après une séquence d'actions. À chaque feuille correspond ainsi un coût total de la réparation qui mène à cette feuille. On peut alors calculer les coûts espérés dans chaque nœud de façon inductive à partir des feuilles : dans un nœud de loterie (résultat d'une observation ou d'une réparation), on calcule l'espérance des coûts de ses nœuds fils alors que, dans un nœud de décision (choix d'une observation ou réparation), on prend comme valeur la valeur du fils avec le plus petit coût espéré. Cette approche permet de résoudre le problème de façon exacte, mais en temps exponentiel en la taille des données car il faut parcourir tout l'arbre pour déterminer le meilleur choix en chaque nœud.

Le fait que cet algorithme exacte est exponentiel n'est pas surprenant : il a été démontré dans [18] que, sauf sous des hypothèses simplificatrices assez fortes (par exemple, absence d'observations et présence d'une unique composante panne sur le système), le problème de *Troubleshooting* est NP-difficile. Cela motive ainsi la recherche d'heuristiques donnant de bonnes solutions pratiques ainsi que d'algorithmes approchés pour le problème de *Troubleshooting*. Il faut cependant noter que, en toute généralité, le problème de *Troubleshooting* est aussi NP-difficile à résoudre dans un sens approché, comme démontré dans [14].

1.5.2 Algorithme exacte sous hypothèses simplificatrices

Sous des hypothèses simplificatrices assez restrictives, il est connu [7,8,14,18] qu'il est possible de résoudre le problème de *Troubleshooting* en temps polynomial. Plus précisément, on suppose que :

- il n'y a qu'une seule composante présentant un défaut ;
- les coûts des réparations sont indépendants ;
- la seule observation possible est celle de l'état du dispositif, o_0, qui a un coût 0.

À travers le réseau Bayésien décrivant le dispositif, on dispose, pour tout $i \in [\![1, n]\!]$, de la probabilité p_i que la réparation r_i résout le problème. L'algorithme polynomial consiste alors 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.

Ces hypothèses 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 [4] pour les problèmes d'élicitation décrit plus en détail ci-après.

1.5.3 Approche myope

Les articles [7, 8] présentent un algorithme heuristique pour le problème de *Troubleshooting* qui fait des hypothèses moins restrictives que les précédentes. 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 leur 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 $a = (o_i, r_j)$ est

$$C^{or}(a, E) = C(o_i) + P(o_i \neq \text{normal} \mid E)C(r_j),$$

où E représente les informations dont on dispose, et on définit $C^{or}(a, E) = C(r_j)$ dans le cas où $a = r_j$ représente une réparation d'une composante non-observable. 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.

À la fin de l'algorithme, cette heuristique basée sur les observations-réparations donnera une séquence $a_1, \ldots, a_k$ d'actions à prendre dans l'ordre, où chaque action a_i représente la réparation d'une composante non-observable, la paire observation-réparation d'une composante observable ou l'appel au service. Le coût espéré de réparation associé à cette séquence d'actions calculée à partir les informations initiales E, $\text{ECR}(E)$, est alors donné par la formule

$$\begin{aligned}\text{ECR}(E) = {} & C^{or}(a_1, E) \\ & + P(o_0 \neq \text{normal} \mid E_1)C^{or}(a_2, E_1) \\ & + P(o_0 \neq \text{normal} \mid E_2)C^{or}(a_3, E_2) \\ & + \cdots \\ & + P(o_0 \neq \text{normal} \mid E_{k-1})C^{or}(a_k, E_{k-1}),\end{aligned}$$

où, pour $j \in \{1, \ldots, k-1\}$, E_j représente les informations à la fin de l'étape j, c'est-à-dire E rajouté des informations que les composantes correspondant aux actions $a_1, \ldots, a_j$ ont été réparés.

La contribution principale de [7, 8] est une autre approche heuristique qui, par rapport au cas précédent, 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. Cela donne ainsi l'espérance de coût myope suite à l'observation o_i, $\text{ECO}(o_i)$, donnée par

$$\text{ECO}(o_i, E) = C(o_i) + \sum_{j=1}^{n_i} \text{ECR}(E \cup \{o_i = j\})P(\{o_i = j\} \mid E),$$

où l'on considère que l'ensemble des résultats possibles de l'observation o_i est $\{1, \ldots, n_i\}$. À chaque étape, on compare ces espérances de coût $\text{ECO}(o_i, E)$ (une pour chaque observation globale) avec l'espérance de coût sans observation $\text{ECR}(E)$ (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. On remarque aussi que la différence entre $\text{ECO}(o_i, E)$ et $\text{ECR}(E)$ peut être vue comme la valeur espérée de l'information myope apportée par l'observation o_i, dans le même sens que la formule de EVOI (1) de la Section 1.3.

Les travaux [7, 8] 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.

1.5.4 Extensions de l'approche myope

Des extensions de l'approche de [7, 8] ont été proposées en particulier dans [11, 13] où les méthodes développés ont permis d'obtenir des résultats assez efficaces pour des cas plus généraux. Plus spécifiquement, l'article [11] considère, d'abord, que les composantes et les actions de réparation ne sont plus en correspondance univoque et que chaque action peut traiter les composantes associées avec une certaine probabilité. Ainsi, les actions ne sont plus parfaites et ne conduisent pas toujours vers une réparation d'une composante. Par ailleurs, on suppose que chaque action a la possibilité de dépanner plusieurs composantes et, réciproquement, chaque composante peut être réparée par des actions différentes. De plus, cet article propose une approche qui améliore la technique myope décrite ci-dessus.

Quant à l'article [13], l'auteur y considère un cas encore plus général où chaque composante est constituée de sous-composantes qui peuvent elles-mêmes être à l'origine de la panne du dispositif et qui peuvent être réparées. En outre, les composantes, vues comme des ensembles de sous-composantes, ne sont pas forcément deux-à-deux disjointes. En conséquence, dans ce modèle il est possible qu'une sous-composante X soit partie de deux composantes différentes, c_i et c_j, $i \neq j$. Selon des résultats de simulations numériques de [13], les algorithmes y développés retournent des stratégies pour le *troubleshooting* beaucoup plus efficaces que l'approche myope pour des problèmes concrets. En effet, pour les modèles considérés, les techniques de [11,13] retournent des solutions dont le coût espéré de réparation a un écart relatif moyen de 2,51% par rapport à l'optimum trouvé par une recherche exhaustive, au lieu de 21,5% pour l'approche myope.

1.6 Élicitation

1.6.1 Élicitation dans le *Troubleshooting*

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. Les articles [3, 4] présentent le problème d'élicitation des fonctions d'utilité et donnent 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 $(x_1, \ldots, x_d)$ et on regarde $u(R)$ comme $u(x_1, \ldots, x_d)$. Les articles [3, 4] supposent alors que u satisfait une hypothèse d'*indépendance additive généralisée*, ce qui permet de l'écrire comme combinaison linéaire de fonctions $u_1, \ldots, u_p$, chacune dépendant seulement d'une partie des caractéristiques x_i, par exemple

$$u(x_1, x_2, x_3, x_4) = \lambda_1 u_1(x_1) + \lambda_2 u_2(x_2, x_4) + \lambda_3 u_3(x_3, x_4). \tag{2}$$

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.

1.6.2 Approche Bayésienne

Afin d'avancer vers le problème d'élicitation, il faut décider comment on pourrait représenter les incertitudes sur l'utilité d'un utilisateur en particulier. L'article [5] utilise une hypothèse centrale : étant donnée une conséquence O d'une décision D, on considère que son utilité u_O est une variable aléatoire qui suit une loi de probabilité $P(u_O)$. L'article suppose empiriquement que la loi P est une mixture de gaussiennes, possiblement tronquées. Lorsque l'on dispose des statistiques assez significatives, on aura la possibilité de reconstruire sa densité de probabilité pour un utilisateur générique. Il faut cependant adapter la loi pour chaque nouvel utilisateur pour prendre en compte les différences de leurs utilités.

Pour faire cette adaptation, on pose des questions d'élicitation, dont les réponses mettent à jour la loi de probabilité de l'utilité. Bien que ces questions nous donnent des informations, on ne doit pas trop en poser car l'utilisateur peut se fatiguer de leur répondre. Par conséquent, il est nécessaire de déterminer un critère pour choisir quelles questions il vaut mieux poser. L'article [5] définit de manière assez similaire qu'à la Section 1.3 une notion de *valeur de l'information* utilisée pour déterminer la question suivante à poser. Cependant, on remarque que cette approche est "*myope*" car on ne considère que la valeur de l'information locale de chaque question. Une technique plus générale consisterait à observer toutes les combinaisons possibles que l'on pourrait construire à partir de questions prises en compte, mais il est bien évident que, pour la grande majorité de cas, une telle technique est intraitable à cause du nombre des combinaisons possibles qui augmente trop vite. C'est pourquoi l'approche proposée dans [5] est une solution assez efficace afin de traiter ce problème.

Par ailleurs, l'article [1] fournit une extension de [5] qui approfondit l'idée de considérer la valeur d'utilité comme une variable aléatoire. En effet, il propose de modéliser le problème d'élicitation par un *processus de décision markovien partiellement observable* (POMDP pour *Partially Observed Markov Decision Process* en anglais) qui est une généralisation des *processus de décision markoviens* (MDP pour *Markov Decision Process* en anglais) où des chaînes des Markov simples sont remplacées par des chaînes cachées. L'article [1] suppose plus précisément que l'ensemble d'états du système modélisé est U, l'ensemble de toutes les fonctions d'utilité possibles, alors que l'ensemble d'observations est tout simplement l'ensemble de toutes les densités de probabilité définies sur U. De plus, le système est capable d'effectuer deux types d'actions : poser des questions et prendre une décision. Ensuite, des méthodes particulières développées pour POMDP sont utilisées afin de résoudre le problème d'élicitation posé au début.

Les techniques proposées dans [1, 5] ont été développés dans [3, 4], comme décrit dans la Section 1.6.1, pour des fonctions d'utilité satisfaisant l'hypothèse d'indépendance additive généralisée (voir (2)). Ces articles montrent comment exploiter une élicitation myope utilisant la notion de *valeur d'information* et proposent un moyen graphique pour représenter les relations d'indépendance.

Une autre extension possible de [1,5] est d'utiliser plutôt des *ensembles optimaux de recommandations* pour choisir quelles questions poser au lieu de les déterminer de façon séquentielle. Cette idée a été explorée, par exemple, dans les articles [16, 17], et consiste à déterminer un ensemble d'alternatives "convenables" parmi celles proposées.

Plus précisément, étant donné un décideur avec sa propre fonction d'utilité, on doit trouver les m meilleures solutions parmi $\alpha_1, \alpha_2, \ldots, \alpha_n$, où $m < n$. L'approche la plus simple consiste à trier les stratégies α_i en ordre décroissante de leurs utilités espérées et prendre ensuite les m premières décisions de la liste triée. En pratique, cette approche n'est pas toujours suffisante [16] car dans la grande majorité de cas on ne connaît l'utilité du décideur qu'avec des incertitudes. Ce serait alors possible de choisir m solutions trop similaires, en particulier si n est relativement grand. Par exemple, supposons que l'on gère un magasin de livres en ligne et que l'on cherche un ensemble de 10 livres à montrer à un utilisateur particulier qui seraient les plus susceptibles de l'intéresser. Supposons que l'on a élicité son utilité et que l'on lui propose les 10 meilleures alternatives selon cette utilité. Comme chaque livre a presque toujours des éditions différentes, il est souhaitable de ne montrer à cet utilisateur qu'une seule proposition parmi ces plusieurs éditions.

Bien que l'on puisse introduire un critère de similarité selon lequel l'on filtre des alternatives, une approche différente a été proposée dans l'article [16] : on définit plutôt un critère pour chaque sous-ensemble de $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ et on cherche ensuite un sous-ensemble qui maximisera ce critère. Ce sous-ensemble est appelé *ensemble optimal de recommandations* (de *Optimal Recommendation Set* en anglais). On note que cette technique assure une *diversité* des alternatives dans l'ensemble trouvé, ce qui permet de proposer à l'utilisateur des bonnes solutions malgré les incertitudes de son utilité. En plus, le choix de l'utilisateur dans cette ensemble contient aussi des informations sur ses préférences, ce qui peut alors être utilisé pour améliorer des connaissances sur sa fonction d'utilité, comme détaillé dans [17], qui relie les ensembles optimaux de recommandations avec les ensembles de questions à choix multiples maximisant la valeur espérée de l'information.

Pour le problème de *Troubleshooting*, il se peut que l'on ne connaisse pas les prix des observations ou réparations de façon exacte, ce qui correspond à ne pas connaître de façon précise la fonction d'utilité. Il est alors important d'être capable de prendre en compte ces incertitudes pour la recommandation d'actions et éventuellement recommander plusieurs actions à la fois pour que l'utilisateur puisse choisir celle qu'il trouve plus convenable. Les techniques de [16, 17] peuvent alors être utiles pour donner une liste d'actions diverses qui pourra en plus, à partir du choix de l'utilisateur, d'avoir des informations supplémentaires sur sa fonction d'utilité.

1.6.3 Approche ensembliste

L'article [10] utilise une autre approche pour la représentation des incertitudes lors de l'élicitation, basée sur des ensembles plutôt que des lois de probabilité. Les auteurs y considèrent un problème légèrement différent du nôtre, consistant à déterminer l'utilité d'un ensemble d'objets $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$, chaque objet étant décrit par n attributs $x_1, x_2, \ldots, x_n$. L'hypothèse faite dans l'article est que l'utilité d'un objet est une combinaison linéaire de fonctions de chacun de ses attributs, c'est-à-dire, l'utilité de l'objet $o = (x_1, x_2, \ldots, x_n)$ est donnée par

$$u(o, w) = \sum_{i=1}^{n} w_i f_i(x_i),$$

où les fonctions $f_1, f_2, \ldots, f_n$ sont connues mais les poids $w_1, w_2, \ldots, w_n$, avec $w_i \geq 0$ pour tout $i \in \{1, \ldots, n\}$, sont à déterminer. Plutôt que déterminer les valeurs précises des w_i, l'information disponible sur ces coefficients est représentée par un ensemble P auquel on est sûr qu'ils appartiennent. Cet ensemble est alors mis à jour lors que des nouvelles informations sont acquises.

Sans perte de généralité et sans aucune autre information a priori sur les w_i, P est défini initialement par

$$P = \left\{ (w_1, \ldots, w_n) \in \mathbb{R}^n \;\middle|\; w_i \geq 0 \text{ pour tout } i \in \{1, \ldots, n\} \text{ et } \sum_{i=1}^{n} w_i = 1 \right\}.$$

On considère que les valeurs qui seront prises par $w_1, \ldots, w_n$ correspondent au centre de P. Pour améliorer les connaissances des w_i, et par conséquent de la fonction u, on cherche à diminuer P à chaque étape. Pour cela, on propose itérativement à l'utilisateur de comparer des paires d'objets, mettant à jour à chaque fois la région P et recalculant les w_i en prenant le centre de P actualisé.

On cherche à minimiser l'hypervolume de P afin que les poids soient déterminés avec la plus petite erreur possible. Ainsi, les comparaisons entre objets (o_i, o_j) présentées à l'utilisateur sont déterminées pour rétrécir le plus la région P. On boucle ce processus jusqu'à ce que le critère de terminaison soit vérifié, par exemple après un nombre maximal de questions posées à l'utilisateur ou lorsque la taille de P est suffisamment petite. L'article [10] présente une implémentation de ces idées, avec une description détaillée du choix des objets à comparer à partir de considérations géométriques. L'algorithme nécessite aussi le calcul des centres des régions P à chaque étape, ce qui est un problème non-trivial.

Une autre approche pour ce même problème est celle du regret minimax, décrit par exemple dans [2]. Le principe reste celui de définir initialement un ensemble P pour les coefficients $(w_1, \ldots, w_n)$ de la même façon et de chercher à poser des questions à l'utilisateur de façon à réduire la taille de P. L'idée du regret minimax est de, à chaque fois, déterminer l'objet o_* qui minimise le regret maximal par rapport à tous les autres objets et tous les éléments de P, soit

$$o_* = \operatorname*{argmin}_{o \in \mathcal{O}} \max_{q \in \mathcal{O}} \max_{w \in P} u(q, w) - u(o, w).$$

On choisit alors son pire adversaire q_*, celui qui maximise son regret, soit

$$q_* = \operatorname*{argmax}_{q \in \mathcal{O}} \max_{w \in P} u(q, w) - u(o_*, w),$$

et on demande à l'utilisateur de comparer o_* et q_*. La réponse de l'utilisateur donne ainsi une inégalité entre $u(q_*, w)$ et $u(o_*, w)$, qui est utilisée pour réduire la région P. On itère jusqu'à ce que P soit suffisamment petit ou que le regret soit nul.

1.7 Objectifs du projet

Pour ce projet, on commence par la 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 [7, 8]. Cette implémentation suppose une connaissance parfaite des paramètres du problème, à savoir les probabilités du réseau Bayésien et les coûts de chaque observation et réparation. La deuxième partie du projet cherche à s'affranchir, au moins partiellement, de ces hypothèses simplificatrices, en supposant par exemple que les coûts ne sont pas parfaitement connus. Notre objectif sera alors d'utiliser des méthodes d'élicitation pour réduire les incertitudes sur les données du problème.

Mots-clés : Troubleshooting, Value of Information, Elicitation.

Références

[1] C. Boutilier. A POMDP formulation of preference elicitation problems. In *Eighteenth National Conference on Artificial Intelligence*, page 239–246, USA, 2002. American Association for Artificial Intelligence.

[2] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9):686–713, 2006.

[3] D. Braziunas and C. Boutilier. Local utility elicitation in gai models. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 42–49, Arlington, Virginia, USA, 2005. AUAI Press.

[4] D. Braziunas and C. Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4):79, dec 2008.

[5] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 363–369. AAAI Press, 2000.

[6] C. Gonzales, L. Torti, and P.-H. Wuillemin. aGrUM: A graphical universal model framework. In *Advances in Artificial Intelligence: From Theory to Practice*, pages 171–177. Springer International Publishing, 2017.

[7] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. Technical report, Microsoft Research Technical Report MSR-TR-94-07, 1994.

[8] D. Heckerman, J. S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, mar 1995.

[9] R. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[10] V. S. Iyengar, J. Lee, and M. Campbell. Evaluating multiple attribute items using queries. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, EC '01, pages 144–153, New York, NY, USA, 2001. Association for Computing Machinery.

[11] 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.

[12] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer New York, New York, NY, 2007.

[13] H. Langseth. Decision theoretic troubleshooting of coherent systems. *Reliability Engineering & System Safety*, 80(1):49–62, apr 2003.

[14] V. Lín. Decision-theoretic troubleshooting: Hardness of approximation. *International Journal of Approximate Reasoning*, 55(4):977–988, jun 2014.

[15] D. North. A tutorial introduction to decision theory. *IEEE Transactions on Systems Science and Cybernetics*, 4(3):200–210, 1968.

[16] B. Price and P. R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, page 541–548. AAAI Press, 2005.

[17] P. Viappiani and C. Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2352–2360, Red Hook, NY, USA, 2010. Curran Associates Inc.

[18] M. Vomlelová. Complexity of decision-theoretic troubleshooting. *International Journal of Intelligent Systems*, 18(2):267–277, jan 2003.

[19] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, N.J., third edition, 1953.