

Rapport SEME - Génération de scénarios pour les véhicules autonomes

Valérie Garcin, Nicoletta Prencipe, Suzanne Schlich, Dorine Tabary

November 4, 2020

1 Présentation du problème

1.1 Présentation par IRT - SystemX

Dans le cadre de la conduite autonome, un scénario est décrit par un ensemble de paramètres du véhicule autonome et de son environnement comme à titre d'exemple: voie rapide, vitesse élevée, rabattement à droite, soleil etc.

Le travail des experts a permis d'identifier des paramètres pouvant constituer des scénarios. L'objectif du sujet proposé consiste à élaborer une méthode mathématique permettant de générer les 1000 scénarios plus pertinents obéissant à de multiples nécessités. Un scénario est un ensemble de paramètres obéissant à certaines nécessités. Ces nécessités ont été définies, avec notamment des règles logiques entre paramètres (par exemple une journée ensoleillée et un temps nuageux ne peuvent pas coexister dans un même scénario), ou même l'élaboration de pondération provenant de deux paramètres (criticité et probabilité). Générer les scénarios sans travail préalable entraîne une explosion combinatoire de solutions. L'objectif de notre travail de recherche est d'utiliser les outils mathématiques afin de fournir un ensemble de solution pertinentes.

Après avoir expliqué l'intérêt de ce problème dans l'industrie, un petit jeu de données sera fournie pour tester et illustrer les solutions proposées.

Les données fournies par l'entreprise IRT-SystemX sous la forme de tableau Excel sont

- d'une part la liste des risques possibles 45 **Features** à inclure dans un scénario avec deux niveaux de **criticité** possibles (A ou B) et cinq niveaux de **probabilité** (A: Very High, B: High, C: Regular, D: Low, E: Very Low).
- d'autre part un matrice donnant les liens d'implications pondérés entre les différents risques possibles.

2 Notre approche au problème

On cherche à trouver parmi les 2^{45} scénarios envisageables 1000 scénarios pertinents. Par pertinents on entend des scénarios prenant en compte les règles logiques entre les paramètres et qui sont intéressants en terme de criticité et probabilité. Ce problème a déjà été soulevé dans [1], avec

l'élaboration d'une approche utilisant la criticité et la probabilité. Dans cette approche, le front de Pareto est un critère de choix de scénarios et permet ainsi de diminuer l'explosion combinatoire.

Notre approche enrichit ce travail sur plusieurs points :

1. une analyse des données fournies ;
2. un filtrage utilisant les liens logiques ;
3. une alternative de l'article [1] concernant la sélection des scénarios ;
4. une implémentation de notre travail.

2.1 Les données

- **Criticité et probabilité** Aux 45 paramètres pouvant définir les scénarios sont associés des niveaux de criticité et de probabilité. Cependant ces niveaux ne nous ont pas été transmis sous forme de valeur mais de lettres telles que mentionnées plus haut. Nous avons donc du faire un choix de valeur pour chaque niveau. Afin de ne pas assumer d'informations que nous n'avions pas, nous avons réparti les valeurs de probabilité d'un critère de façon homogène entre 0 et 1, c'est à dire en choisissant : $A = 0.9, B = 0.7, C = 0.5, D = 0.3, E = 0.1$, et les valeurs de criticité d'un critère avec $A = 1, B = 0, 5$.

- **Implications**

Il nous a été fourni une matrice de taille 45×45 représentant les liens entre les différents paramètres sous forme de réels entre 0 et 1. Notre démarche s'inscrivant dans une approche utilisant des liens logiques nous avons décidé d'en extraire une matrice d'implication en ne gardant que les valeurs égales à 1. Ainsi la matrice permet de lire des implications de la façon suivante : le critère i implique le critère j si et seulement si l'entrée (j, i) de la matrice est 1. Les autres entrées de la matrice sont nulles et indiquent qu'il n'y a pas à priori de liens logiques entre ces deux paramètres. Par exemple, le paramètre 21 "speed bump" (dos d'âne) implique le paramètre 6 "small obstacle" (petit obstacle), ce qui est noté par un 1 dans la case $(6, 21)$ de la matrice.

- **Exclusions**

Dans un deuxième temps, nous avons décidé d'enrichir les données d'implications logiques entre les paramètres qui nous étaient fournis par des liens logiques sous forme d'exclusion entre les paramètres. Ces nouveaux liens prennent la forme d'une matrice d'exclusion (symétrique) de taille 45×45 dont l'entrée (i, j) vaut 1 si et seulement si les paramètres i et j s'excluent mutuellement dans un scénario. Par exemple, les paramètres *pont* et *rond-point* ne peuvent pas coexister dans un scénario.

Cependant, notre travail n'étant pas de construire une base de données exhaustive des exclusions possibles, nous nous sommes limités à quelques exclusions, afin de pouvoir travailler avec ces notions, et tout en laissant le soin aux experts de remplir soigneusement ces données à posteriori pour une utilisation rigoureuse.

2.2 Analyse des outils

Présentation des outils En recherche opérationnelle, un problème est modélisé à travers des variables, des contraintes sur lesquelles sont construites ces variables et une fonction objectif.

Il existe deux grandes **heuristiques** : la programmation linéaire et la programmation par contrainte.

Comme son nom l'indique **la programmation linéaire** (*PL*) permet d'optimiser (maximiser ou minimiser) une fonction linéaire de plusieurs variables qui sont reliées par des relations linéaires appelées contraintes.

La **Programmation par Contraintes** (*PPC*) modélise le problème par des variables de décision et de contraintes. Une contrainte est une relation entre une ou plusieurs variables qui limite les valeurs que peuvent prendre simultanément chacune des variables liées par la contrainte.

Notre choix Afin de nous adapter aux contraintes liées au format de la SEME (notamment le temps court et l'objectif de résultat), aux contraintes liées à l'entreprise (notamment la confidentialité des données, les caractéristiques réalistes du problème proposé), nous avons privilégié la programmation par contraintes.

De plus, la programmation par contraintes est dotée d'un panel d'outils déjà existants, simples, efficaces et adaptés à la problématique.

Ainsi, Choco-solver [2] est une librairie java libre, openSource. Elle a été développée par l'École des Mines de Nantes.

Elle décompose la résolution en trois étapes :

1. Modelisation ;
2. Résolution ;
3. Analyse (Parsing).

2.3 Implémentation de la méthode

2.3.1 Formalisation du problème

Le problème est défini par un triplet (χ, D, C) .

Un des défis fut de transformer les règles issues de la réalité en contraintes du solver par contraintes. Ces règles furent dans un premier temps définies mathématiquement dans l'équation (1), avec la fonction préliminaire C^* .

$$C^* = \{\overline{\chi_i \wedge \chi_j}\}_{i \neq j \in I \times J} \bigcup \{\overline{\chi_m \wedge \chi_n}\}_{m \neq n \in M \times N} \bigcup \{\pi_t(s) \geq \alpha\} \quad (1)$$

où

- I, J, K, L, M et N des ensembles d'indices de variables ou paramètres ;
- $s = \{\chi_l\}_{l \in L}$, avec L ensemble des indices des variables ou paramètres retenus pour un scénario;
- π_t est la mesure sur χ définie par l'équation (5) ;
- α représente un seuil, permettant de filtrer les scénarios retenus.

Ce travail préalable a permis par la suite d'implémenter ces règles comme des contraintes de la programmation par contraintes. Par exemple, la règle réelle associée au fait qu'il ne peut exister

de scénarios avec le paramètre brouillard et le paramètre pluie en même temps est incorporée dans une contrainte d'exclusion.

$$\begin{cases} \chi = \{\chi_i = \text{"feature"}, i \in \{1, \dots, 45\}\} \\ D = \{D_i = \{0, 1\}, i \in \{1, \dots, 45\}\} \text{ est l'ensemble des domaines des variables,} \\ \mathcal{C} = \{\mathcal{C}_i = (\{\chi_{i,k}\}_{k \in K}, \mathcal{R}_i)\} = \{C_1, C_2, C_3\} \end{cases} \quad (2)$$

où

- χ est l'ensemble de variables ou paramètre du problème,
- D est l'ensemble des domaines des variables,
- \mathcal{C} est l'ensemble de contraintes issues de C^* de l'équation (1), avec
 - C_1 la contrainte d'implication, $C_1 = \{\overline{\chi_i \wedge \chi_j}\}_{\{i,j\} \in \mathcal{E}}$ avec \mathcal{E} l'ensemble des paires $\{i, j\}$ de paramètres exclus.
 - C_2 la contrainte d'exclusion, $C_2 = \{\overline{\chi_i \wedge \chi_j}\}_{(i,j) \in \mathcal{I}}$ avec \mathcal{I} l'ensemble des couples (i, j) tel que $i \rightarrow j$.
 - C_3 la contrainte de seuil, $C_3 = \{\pi_t(s) \geq \alpha\}$ avec $s = \{\chi_l\}_{l \in L}$, L ensemble des indices des variables ou paramètres retenus pour un scénario, et π_t est la mesure sur χ définie par l'équation (5).

2.3.2 Filtrage

La première étape de la méthode de programmation par contraintes consiste à réaliser une diminution conséquente de l'ensemble des scénarios par filtrage. Le filtrage est réalisé par l'implémentation dans Chocosolver de deux types de contraintes logiques :

- l'exclusion, $\overline{A \wedge B}$,
- l'implication, $\overline{A \wedge \overline{B}}$.

2.3.3 Pertinence des scénarios

Le deuxième objectif dans ce projet de génération de scénario est de produire des scénarios qui soient probables de se produire ainsi que des scénarios qui soient critiques. Notons P et C les fonctions qui associent à chaque critère respectivement leur probabilité et leur criticité comme présenté dans la partie 2.1 de ce rapport. On définit à partir de P et C une notion de probabilité d'un scénario s ainsi qu'une notion de criticité d'un scénario s , soit :

$$\text{Proba}(s) = \prod_{i \in s} P(i) \prod_{i \notin s} (1 - P(i)) \quad (3)$$

$$\text{Crit}(s) = \frac{1}{45} \sum_{i \in s} C(i) \quad (4)$$

On peut imaginer que suivant l’usage, l’utilisateur ait besoin de sélectionner des scénarios surtout probable ou surtout critique. Ainsi, on se propose de définir une mesure à la fois de la probabilité et de la criticité d’un scénario en pondérant ces deux indicateurs par un paramètre $t \in [0, 1]$. On considère :

$$\pi_t(s) = t\text{Crit}(s) + (1 - t)\text{Proba}(s) \quad (5)$$

Ainsi une valeur de $t = 0$ donne une fonction π_0 qui ne mesure que la probabilité qu’un scénario se produise et à l’opposé une valeur de $t = 1$ donne une fonction π_1 qui ne mesure que la criticité d’un scénario. Le choix est donc laissé à l’utilisateur de régler le paramètre t à sa convenance. Notons par ailleurs que comme $\text{Proba}(s), \text{Crit}(s) \in [0, 1]$, on a également $\pi_t(s) \in [0, 1], \forall s$ et $\forall t \in [0, 1]$.

La deuxième partie de l’implémentation de notre méthode est donc de fixer une valeur seuil $\alpha \in [0, 1]$ et de filtrer les scénarios s tels que $\pi_t(s) \geq \alpha$.

Après ces deux étapes, on ne retient donc que les scénarios logiquement pertinents dont la valeur de π_t est supérieure à notre valeur seuil α .

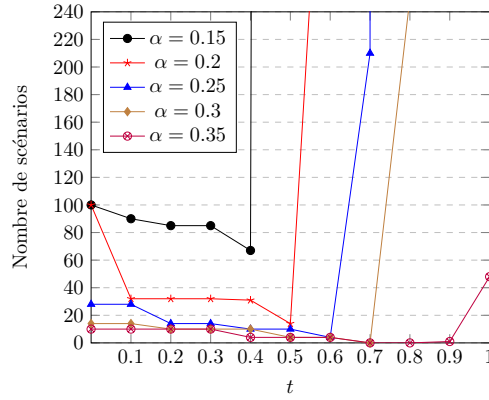
2.3.4 Résultats

Une fois le code implémenté, le résultat obtenu sort sous deux formats : directement dans le terminal de commande et dans un fichier.txt, comme illustré dans la capture d’écran de la Figure 1.

Afin d’analyser le comportement de la solution proposée, nous avons réalisé une série de tests avec les variables t et α . Elles sont détaillées dans la section 2.3.3 de notre rapport. Le résultat est le nombre de scénarios proposé par le PPC. L’ensemble des résultats des différents tests est répertorié dans l’annexe 1. Comme 1000 scénarios nous a été demandé, nous nous sommes concentrées sur l’analyse des résultats donnant un nombre de solutions inférieurs à 1000. Une valeur très supérieure à 1000 résultats correspond à un x dans ce tableau.

La figure 1a illustre le comportement de notre approche en prenant en compte le nombre de scénarios en fonction de t . Les courbes sont analysées selon cinq valeurs de α . Tout d’abord, on observe une redondance du nombre de scénarios. Un résultat à 100 scénarios apparaît par exemple 9 fois. Une première explication réside dans le fait que des valeurs de $\pi_t(s)$ peuvent être proches même avec des t et α différentes. Une seconde explication serait que des scénarios différents avec des $\pi_t(s)$ voisins pourraient franchissent ensemble la valeur seuil α .

Un deuxième comportement intéressant apparaît dans la figure 1a. On observe une diminution du nombre de scénarios jusqu’à un certain t , puis ensuite une augmentation drastique. Par exemple, pour $\alpha = 0.15$, le nombre de scénarios diminue jusqu’au nombre de 45 pour un t égale à 0.4, et augmente ensuite. Une explication réside sur l’utilisation de deux critères (criticité et probabilité) pondérés par t . Une variation de t attribue plus ou moins d’importance à un type de critères, d’où l’apparition de deux comportements différents (courbe décroissante puis croissante).



(a) Nombre de scénarios en fonction de t .

3 Pour aller plus loin

3.1 Enrichissement en terme de données

Pour une implémentation rigoureuse de la programmation par contrainte dans Choco, il apparaît important de se pencher sur les données du problème et notamment:

- Les contraintes du problème doivent être aussi nombreuses et pertinentes que possible. Le regard d'un expert en sécurité routière par exemple pourrait s'avérer riche en enseignement. Plus les matrices des implications et des exclusions sont fournies, plus le filtrage de départ est important.
- L'analyse des paramètres et leur regroupement par typologie seraient un plus : météo, voirie, véhicule autonome, véhicule rencontré, état du conducteur, ...
- L'obtention de valeurs précises quant aux probabilités et non des plages de valeurs affinerait le choix des scénarios.
- Une criticité des paramètres plus étendue permettrait également un choix plus fin.

3.2 Autres pistes possibles

- Changer la fonction π pour modifier la notion de pertinence utilisée.
- La donnée d'une matrice de covariance entre les différents paramètre pourrait permettre une approche probabiliste complètement différente quant à la génération des scénarios.
- Une autre amélioration possible est le perfectionnement du filtrage des données en ajoutant un paramètre tenant compte du niveau d'autonomie du véhicule.

4 Conclusion

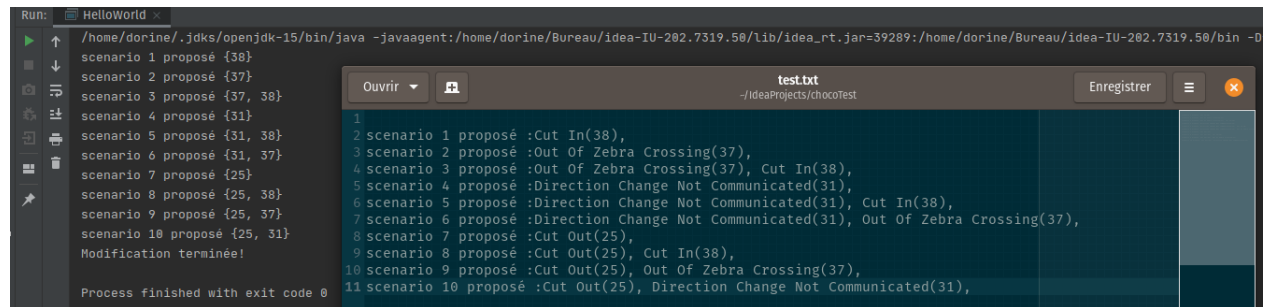
Confrontées à un problème d'une entreprise, nous avons pu échafauder une nouvelle stratégie afin de fournir une solution ajustée au mieux au paradigme industriel. Ce problème s'est en effet révélé

un réel défi qui a nécessité de nombreuses adaptations, que ce soit au niveau des données fournies issues du monde réel, ou au niveau du type de solution demandée en passant par la méthode de résolution.

Les différentes compétences de notre équipe ont permis d'étayer une analyse poussée du problème. Une solution fut rapidement fournie pour être ensuite améliorée au cours de ces trois jours de travail en commun. L'utilisation conjointe de notre savoir universitaire a servi de base à une effusion d'idées différentes renforçant par ce travail de groupe, l'enrichissement intellectuel déjà conséquent issu de l'utilisation dans un cadre entrepreneurial d'outils mathématiques.

Le problème étudié ouvre de nombreuses portes vers de futurs travaux de recherche. Une de ces pistes serait de lier la génération de scénarios à la thématique du choix social. Ainsi, conducteur, constructeur et assurance peuvent avoir des avis divergents sur la pertinence d'un scénario. Par exemple, lors de l'élaboration d'un scénario, le conducteur (peu conciliant) donne beaucoup d'importance à la criticité et oublie le paramètre vitesse. Au contraire, les aspects probabilistes et vitesses sont primordiaux pour les assurances. Enfin, le risque de piratage est central pour le constructeur. Définir les différentes possibilités, et un système de vote contentant tous les acteurs améliorerait l'adaptation de la solution proposée.

5 Annexes



```

Run: HelloWorld
/home/dorine/.jdk/openjdk-15/bin/java -javaagent:/home/dorine/Bureau/idea-IU-202.7319.50/lib/idea_rt.jar=39289:/home/dorine/Bureau/idea-IU-202.7319.50/bin -D
scenario 1 proposé {38}
scenario 2 proposé {37}
scenario 3 proposé {37, 38}
scenario 4 proposé {31}
scenario 5 proposé {31, 38}
scenario 6 proposé {31, 37}
scenario 7 proposé {25}
scenario 8 proposé {25, 38}
scenario 9 proposé {25, 37}
scenario 10 proposé {25, 31}
Modification terminée!
Process finished with exit code 0

test.txt
~/IdeaProjects/chocoTest
1
2 scenario 1 proposé :Cut In(38),
3 scenario 2 proposé :Out Of Zebra Crossing(37),
4 scenario 3 proposé :Out Of Zebra Crossing(37), Cut In(38),
5 scenario 4 proposé :Direction Change Not Communicated(31),
6 scenario 5 proposé :Direction Change Not Communicated(31), Cut In(38),
7 scenario 6 proposé :Direction Change Not Communicated(31), Out Of Zebra Crossing(37),
8 scenario 7 proposé :Cut Out(25),
9 scenario 8 proposé :Cut Out(25), Cut In(38),
10 scenario 9 proposé :Cut Out(25), Out Of Zebra Crossing(37),
11 scenario 10 proposé :Cut Out(25), Direction Change Not Communicated(31),

```

Figure 1: Sortie obtenue

References

- [1] Ismet Addoui, Tarek Chouaki, Ambrogio Delli Colli. Generating relevant scenarios for intelligent transportation service. 2019 8th International Conference on Transportation and Traffic Engineering, ICTTE 2019, Dec 2019, Auckland, New Zealand. fhal-02351552ff.
- [2] Jussien, Narendra, Guillaume Rochart, and Xavier Lorca. "Choco: an open source java constraint programming library." 2008.

$\alpha \backslash t$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.05	704	x	x	x	x	x	x	x	x	x	x
0.075	318	33102	x	x	x	x	x	x	x	x	x
0.1	225	240	240	21450	x	x	x	x	x	x	x
0.125	169	169	169	169	15468	x	x	x	x	x	x
0.15	100	90	85	85	67	9564	x	x	x	x	x
0.175	100	85	85	32	31	65	9397	x	x	x	x
0.2	100	32	32	32	31	14	490	1873	x	x	x
0.225	30	29	28	14	14	10	10	580	12993	x	x
0.25	28	28	14	14	10	10	4	210	5127	x	x
0.275	28	14	14	10	10	10	4	0	698	40273	x
0.3	14	14	10	10	10	4	4	0	250	6643	x
0.325	14	10	10	10	4	4	4	0	0	201	4899
0.35	10	10	10	10	4	4	4	0	0	1	48
0.4	10	10	10	4	4	0	0	0	0	0	0
0.45	10	4	4	4	0	0	0	0	0	0	0
0.5	4	4	4	4	0	0	0	0	0	0	0
0.55	4	4	4	0	0	0	0	0	0	0	0
0.6	4	4	0	0	0	0	0	0	0	0	0
0.65	4	0	0	0	0	0	0	0	0	0	0
0.7	0	0	0	0	0	0	0	0	0	0	0

Table 1: Tableau des résultats obtenus (nombre de scénarios total).