Association Mathématique du Québec

La description, la représentation et la résolution d'une énigme[1]

Denis Lavigne,
Collège militaire royal de Saint-Jean

INTRODUCTION

Une énigme. Souvent passionnante, rarement lassante, on s'y laisse fréquemment prendre au jeu. On la résout par tâtonnement, intuition, on analyse les indices de façon séquentielle et/ou globale, on découvre les incohérences, les évidences ; tout cela fait partie des éléments qui permettent à une personne de poursuivre la recherche de la solution.

Cet article propose une modélisation mathématique d'une énigme faisant intervenir un certain nombre d'indices, une représentation informatique de celle-ci à l'aide d'un langage de programmation algébrique ainsi qu'une résolution de l'énigme via l'utilisation d'un logiciel spécialisé d'optimisation. La résolution de l'énigme peut être effectuée facilement sur un bout de papier. L'objectif de l'article consiste à développer la curiosité du lecteur envers l'apprentissage et l'utilisation d'un langage de programmation algébrique.

L'intérêt du présent article est multiple :

- il présente une vision nouvelle d'un type de problème bien connu de tous ;
- il présente un cheminement scientifique élégant, naturel et puissant (modélisation, représentation, résolution) dans un contexte surprenant ;
- il constitue un exemple académique amusant et formateur ;
- il constitue un exemple de lien faisant intervenir le langage ordinaire, la représentation symbolique et la représentation informatique ;
- il illustre la simplicité et la puissance d'un langage de programmation algébrique ;
- il mentionne un exemple d'application faisant intervenir ce type de raisonnement.

L'énigme

Une enseignante que je connais bien m'a fait parvenir l'énigme suivante qu'elle a proposée à certains de ses élèves de 6^{e} année. Une énigme similaire, dite d'Einstein, est proposée sur le web (il suffit de

[1] Je remercie les arbitres pour leurs commentaires constructifs qui ont favorisé le développement d'une version finale de qualité.

mentionner *Énigme d'Einstein* dans *www.google.ca* pour obtenir une liste impressionnante de sites traitant de celle-ci).

Cinq maisons alignées et de couleurs différentes sont habitées par des hommes de nationalités et de professions différentes. Chacun a une boisson préférée distincte et un animal domestique également différent des autres. Voici les indices dont vous disposez :

1. Une personne du groupe préfère boire de l'eau.
2. L'Anglais habite la maison rouge.
3. Le chien appartient à l'Espagnol.
4. On boit du café dans la maison verte.
5. L'Ukrainien boit du thé.
6. La maison verte est située immédiatement à droite de la blanche.
7. Le sculpteur élève des escargots.
8. Le diplomate habite la maison jaune.
9. On boit du lait dans la maison du milieu.
10. Le Norvégien habite la première maison à gauche.
11. Le médecin habite la maison voisine de celle où habite le propriétaire du renard.
12. La maison du diplomate est voisine de celle où il y a un cheval.
13. Le violoniste boit du jus d'orange.
14. Le Japonais est acrobate.
15. Le Norvégien habite à côté de la maison bleue.

Et la question, bien sûr : à qui appartient le zèbre ?

Jouez le jeu en complétant le tableau ci-dessous (où le rang va de 1 à 5, de gauche à droite). Une approche intéressante vous sera présentée pour résoudre ce genre de problème et la solution vous sera donnée. Quelle est votre réponse ?

Couleur	Profession	Nationalité	Animal	Boisson	Rang

Tableau 1 : Votre solution de l'énigme

Une approche rigoureuse

Il y a plusieurs façons d'aborder un tel problème. Je propose de débuter en décrivant les ensembles qui sont utilisés dans l'énoncé :

- les couleurs : blanche, rouge, jaune, verte, bleue

- les professions : violoniste, acrobate, sculpteur, médecin, diplomate
- les nationalités : anglais, ukrainien, japonais, espagnol, norvégien
- les animaux : chien, escargots, cheval, renard, zèbre
- les boissons : eau, jus, thé, lait, café
- les rangs : 1 à 5 (de gauche à droite)

Tout au cours de cet article, je ferai le lien entre la représentation mathématique et la représentation algébrique informatique de l'énigme. Voici donc les ensembles tels qu'ils doivent être présentés dans le langage de programmation algébrique MPL[2] :

```
INDEX
  couleur    := (blanche, rouge, jaune, verte, bleue);
  profession := (violoniste, acrobate, sculpteur, medecin, diplomate);
  nationalite:= (anglais, ukrainien, japonais, espagnol, norvegien);
  animal     := (chien, escargots, cheval, renard, zebre);
  boisson    := (eau, jus, the, lait, cafe);
  rang       := (1, 2, 3, 4, 5);
```

Encadré 1 : Représentation des ensembles selon *MPL*

À chaque combinaison est associée une variable binaire. Par exemple, la combinaison *Rouge-Acrobate-Japonais-Cheval-Eau-2* est associée à une variable qui peut être soit vraie, soit fausse (une combinaison est dite vraie si elle respecte chacun des indices). On associe à chacune des combinaisons la variable binaire indicée x_{ijklmn} où i représente *la couleur*, j *la profession*, k *la nationalité*, l *l'animal*, m *la boisson* et n *le rang*. Nous avons donc :

$$x_{ijklmn} = \begin{cases} 1 & \text{si la combinaison } (i,j,k,l,m,n) \text{ est vraie} \\ 0 & \text{si la combinaison } (i,j,k,l,m,n) \text{ est fausse} \end{cases}$$

```
BINARY VARIABLES
  x[couleur, profession, nationalite, animal, boisson, rang];
```

Encadré 2 : Représentation d'une variable indicée selon *MPL*

Résoudre l'énigme consiste à trouver un ensemble de cinq combinaisons valides (donc vraies) qui permettent d'obtenir la solution complète de l'énigme. Un tel ensemble existe si les indices sont cohérents (c'est-à-dire si les indices ne sont pas contradictoires, ce qui mènerait la résolution de cette énigme dans un cul-de-sac).

[2]Mathematical Programming Language de Maximal Software (www.maximalsoftware.com)

Les contraintes implicites

Avant même de songer à modéliser les indices donnés dans l'énoncé du problème, il faut déterminer les contraintes implicites de l'énigme. Par exemple, nous savons qu'il doit y avoir une et une seule maison de chaque couleur. Ainsi, parmi toutes les combinaisons ayant la couleur $i = blanche$, une et une seule doit être vraie (et dont la variable associée doit être égale à 1). Les combinaisons *Blanche-Violoniste-Norvégien-Chien-Jus-3* et *Blanche-Sculpteur-Ukrainien-Escargots-Thé-5* ne peuvent donc pas être vraies simultanément puisque nous aurions alors deux combinaisons utilisant la maison blanche. La somme de toutes les combinaisons ayant $i = blanche$ doit donc être égale à 1. On obtient la contrainte

$$\sum_{jklmn} x_{i=blanche,jklmn} = 1.$$

Puisque le même raisonnement s'applique aux autres couleurs ($i = rouge$, $i = jaune$, ...), on écrit

$$\sum_{jklmn} x_{ijklmn} = 1 \quad \text{où} \quad i = \text{blanche, rouge, jaune, verte, bleue.}$$

L'équation précédente génère un ensemble de 5 contraintes similaires concernant la couleur des maisons. La représentation *MPL* d'un tel ensemble de contraintes est simple (voir Encadré 3). On spécifie tout d'abord le nom de l'ensemble de contraintes. Ici, l'ensemble de contraintes *UneSeuleCouleur* est indicé selon les couleurs, signifiant qu'une contrainte particulière sera générée pour chacune de celles-ci. On définit ensuite la structure de l'ensemble de contraintes. Il s'agit d'une somme sur plusieurs indices (correspondants aux indices j, k, l, m et n ci-dessus) de la variable x (définie à l'Encadré 2) et cette somme doit être égale à 1.

```
UneSeuleCouleur[couleur]:
  Sum(profession, nationalite, animal, boisson, rang: x) = 1;
```

Encadré 3 : Représentation d'un ensemble de contraintes implicites selon *MPL*

Un ensemble de contraintes similaires est nécessaire pour les nationalités. En effet, une personne d'une nationalité donnée doit habiter une et une seule maison. Ainsi, parmi toutes les combinaisons ayant la nationalité $k = anglais$, une et une seule doit être vraie. La somme de toutes les combinaisons ayant $k = anglais$ doit donc être égale à 1. On obtient donc la contrainte

$$\sum_{ijlmn} x_{ijk=anglais,lmn} = 1.$$

Et, encore une fois, puisque le même raisonnement s'applique aux autres nationalités ($k = ukrainien$, $k = japonais, \ldots$), on écrit simplement

$$\sum_{ijlm} x_{ijklmn} = 1 \quad \text{où} \quad k = \text{anglais, ukrainien, japonais, espagnol, norvégien.}$$

Ceci génère un ensemble de 5 contraintes similaires concernant cette fois-ci les nationalités.

De la même façon, nous avons 4 ensembles additionnels de contraintes implicites (pour les professions, les animaux, les boissons et les rangs), ce qui fait un total de $6 \times 5 = 30$ contraintes implicites.

Les contraintes explicites (les indices !)

La modélisation de plusieurs indices est aisée (la plus facile étant celle de l'indice 1 qui ne fait que compléter la description de l'ensemble des boissons). Les indices 2, 3, 4, 5, 7, 8, 9, 10, 13 et 14 peuvent tous être représentés de façons similaires. En effet, chacun de ces 10 indices ne nécessite qu'une contrainte correspondante (pour un total de 10 contraintes). Par exemple, voici la modélisation de l'indice numéro 8 : *le diplomate habite la maison jaune* :

$$\sum_{klmn} x_{ijklmn} = 1 \quad \text{où} \quad i = jaune \text{ et } j = diplomate.$$

```
Indice8:
  Sum(couleur = "jaune", profession = "diplomate",
      nationalite, animal, boisson, rang: x)  =  1;
```

Encadré 4 : Représentation de la contrainte explicite liée à l'indice 8 selon *MPL*

La modélisation des indices 6, 11, 12 et 15 exige une plus grande expérience de la modélisation. Voici la modélisation de l'indice 15 : *le Norvégien habite à côté de la maison bleue* (la modélisation des indices 6, 11 et 12 est similaire à celle-ci).

Afin de modéliser cet indice, il nous faut ajouter deux nouvelles variables binaires indicées chacune sur l'ensemble *rang : rangNorvégien* représentant le rang de la maison du Norvégien et *rangBleue* représentant le rang de la maison bleue. Ces nouvelles variables doivent apparaître suite à la définition de la variable indicée x présentée précédemment. Puisque la cardinalité de l'indice *rang* est égale à 5, l'encadré suivant génère une dizaine de nouvelles variables binaires.

```
rangNorvegien[rang];
rangBleue[rang];
```

Encadré 5 : Représentation de deux nouvelles variables indicées selon *MPL*

Voici les cinq contraintes utilisées afin de représenter l'indice 15 :

$$\begin{aligned}
&\sum_{ijlm} x_{ijklmn} = rangNorvégien_n \quad \text{où } k = Norvégien \quad \text{et } n = 1, \ldots, 5 \text{ ;} \\
&\sum_{jklm} x_{ijklmn} = rangBleu_n \qquad \text{où } i = bleue \qquad \text{et } n = 1, \ldots, 5 \text{ ;} \\
&rangBleue_1 - rangNorvégien_2 \leq 0 \text{ ;} \\
&rangBleue_5 - rangNorvégien_4 \leq 0 \text{ ;} \\
&rangBleue_n - rangNorvégien_{(n-1)} - rangNorvégien_{(n+1)} \leq 0 \quad \text{où} \quad n = 2, 3, 4.
\end{aligned}$$

La première (5 contraintes puisque $n = 1, \ldots, 5$) assure que le rang de la maison habitée par le Norvégien est déterminé par la variable $rangNorvégien$. La deuxième (5 contraintes) est similaire. Les trois autres (1, 1 et 3 contraintes respectivement) modélisent la notion « d'à côté » où une attention particulière doit être accordée aux extrémités.

```
Indice15_1[rang]:
  Sum(couleur, profession, nationalite = "norvegien", animal, boisson: x) = rangNorvegien;

Indice15_2[rang]:
  Sum(couleur = "bleue", profession, nationalite, animal, boisson: x) = rangBleue;

Indice15_3:
  rangBleue[rang:= 1] - rangNorvegien[rang:= 2] <= 0;

Indice15_4:
  rangBleue[rang:= Last(rang)] - rangNorvegien[rang:= Last(rang) - 1] <= 0;

Indice15_5[rang] WHERE ((rang > 1) AND (rang < 5)):
  rangBleue[rang] - rangNorvegien[rang:= rang - 1] - rangNorvegien[rang:= rang + 1] <= 0;
```

Encadré 6 : Représentation des contraintes liées à l'indice 15 selon *MPL*

Ainsi, l'indice 15 nécessite l'ajout de 15 contraintes. Les indices 6, 11, 12 et 15 génèrent donc $4 \times 15 = 60$ contraintes. Le nombre total de contraintes atteint la centaine (30 contraintes implicites + (10 + 60) contraintes explicites).

Définition de la fonction-objectif

La définition des variables et des contraintes étant complétée, il reste à fournir une fonction-objectif à optimiser (*MPL* exige la présence d'une telle fonction). Ici, n'importe quelle fonction-objectif serait acceptable puisqu'il s'agit, en fait, d'un problème de programmation par contraintes (seule la réalisabilité de l'ensemble des contraintes nous intéresse). La fonction-objectif suivante consiste à minimiser le nombre de combinaisons nécessaires à la résolution de l'énigme (la maximisation serait acceptable pour ce problème puisque la fonction-objectif n'est qu'artificielle et que le logiciel d'optimisation ne cherche qu'à respecter les contraintes du problème).

$$\text{Fonction-objectif à minimiser : } \sum_{ijklmn} x_{ijklmn}.$$

```
MIN Sum(couleur, profession, nationalite, animal, boisson, rang: x);
```

Encadré 7 : Représentation de la fonction-objectif selon *MPL*

Le problème ainsi posé constitue un problème d'optimisation linéaire en nombres entiers. Si l'ensemble des contraintes est cohérent, il existe un ensemble de cinq combinaisons qui respectent chacune d'entre elles. Il peut exister plus de cinq combinaisons valides mais il faut en trouver cinq qui fournissent une solution complète de l'énigme. La valeur optimale est connue et est donc égale à 5.

La représentation et la résolution de l'énigme

La représentation du problème est maintenant complétée via le langage de programmation algébrique *MPL*. Il est important de noter que *MPL* ne résout pas les problèmes qu'il permet de représenter (il en est de même pour les autres langages de programmation algébrique). Il fait plutôt appel à un logiciel d'optimisation spécialisé. *MPL* accepte plusieurs des logiciels d'optimisation performants connus sur le marché et le choix est laissé à la discrétion du développeur. Nous utilisons ici le logiciel *CPLEX*[3] de l'entreprise *ILOG*[4] et nous disons donc que nous utilisons la technologie MPL-CPLEX. Ce lien représentation-résolution ne demande aucun effort à l'utilisateur.

MPL effectue la représentation du problème et le résultat de cette étape consiste en une représentation acceptable pour un logiciel d'optimisation tel *CPLEX* qui effectue par la suite son travail d'optimisation.

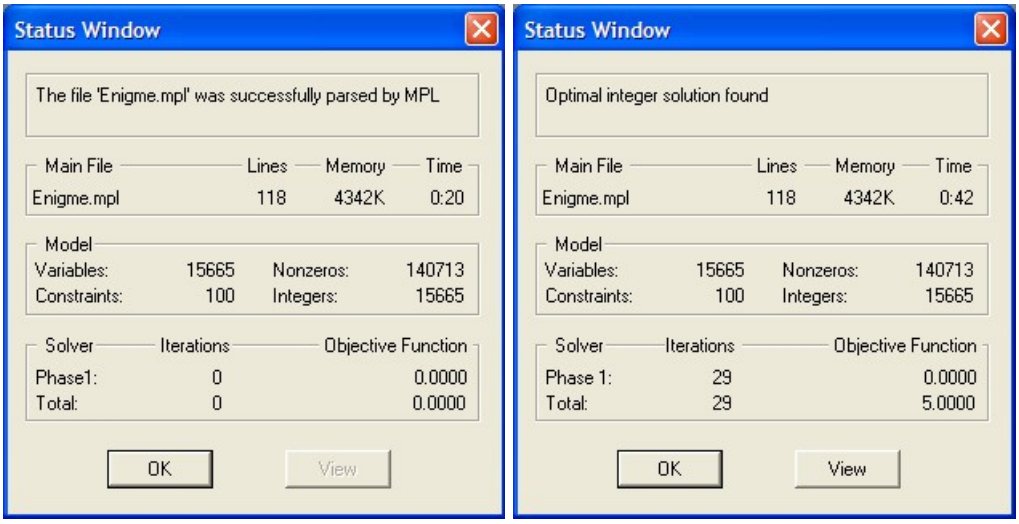

Figure 1 : Résultats de la représentation *MPL* et de l'optimisation *CPLEX*

On y lit entre autres que 0,20 seconde a été nécessaire pour l'opération réussie de représentation et que ce problème possède 15 665 variables entières (que nous savons binaires) et une centaine de contraintes (il est courant en pratique que le nombre de variables dépasse de beaucoup le nombre de contraintes). On y constate aussi qu'une solution optimale entière a été trouvée en 0,42 seconde. Cette résolution a nécessité 29 itérations et la valeur de la fonction-objectif est 5 (tel que prévu). Il est finalement possible de consulter la solution de l'énigme (ainsi que diverses informations habituelles fournies lors de la résolution d'un problème d'optimisation).

[3] www.ilog.fr/products/cplex
[4] www.ilog.fr

```
VARIABLE x[couleur,profession,nationalite,animal,boisson,rang] :

  couleur  profession  nationalite  animal     boisson   rang
 ------------------------------------------------------------
  blanche  violoniste  espagnol     chien      jus         4
  rouge    sculpteur   anglais      escargots  lait        3
  jaune    diplomate   norvegien    renard     eau         1
  verte    acrobate    japonais     zebre      cafe        5
  bleue    medecin     ukrainien    cheval     the         2
 ------------------------------------------------------------
```

Encadré 8 : La solution de l'énigme telle que fournie par *MPL*

La puissance d'un langage de programmation algébrique

Plusieurs connaissent la macro complémentaire *Solveur*[5] de *Frontline Systems* qui est intégrée au logiciel *Excel*[6] de *Microsoft*. Ils l'utilisent afin de résoudre des problèmes d'optimisation. Quoique celle-ci soit efficace pour résoudre des modèles de taille réduite et bien que cette capacité d'*Excel* s'avère un véhicule intéressant pour la formation des élèves lors d'un cours d'introduction à la recherche opérationnelle ou à la modélisation mathématique, il est difficile d'imaginer la gestion d'un modèle comme celui proposé dans cet article autrement que par l'utilisation d'un langage de programmation algébrique.

En effet, ces langages (MPL, AMPL[7], GAMS[8], OPL[9],...) permettent de créer un nombre considérable de variables et de contraintes d'un certain type via l'utilisation judicieuse d'indices (ne pas confondre les indices tel que définis à l'Encadré 1 avec les indices de l'énigme). Par exemple, la variable x proposée à l'Encadré 2 génère à elle seule 15 625 variables (5^6 possibilités puisque chaque indice i, j, k, l, m, n possède 5 valeurs possibles). Il serait lourd de générer, gérer et résoudre un problème de modélisation mathématique et d'optimisation de cette taille via le *Solveur* d'*Excel*. Dès que le nombre de variables et/ou de contraintes augmente de façon appréciable, la simplicité et la puissance d'un langage de programmation algébrique deviennent des atouts indéniables lorsque vient le temps de déterminer l'outil à privilégier pour modéliser et résoudre des problèmes d'optimisation. Par ailleurs, quoique la résolution de l'énigme soit simple et qu'elle ne nécessite pas l'utilisation de logiciels puissants, certaines questions exigent un effort supplémentaire et l'apport de ces outils peut alors s'avérer intéressant.

- La solution proposée est-elle unique ?
- Qu'advient-il si l'on modifie, ajoute ou élimine un indice ?
- Dans le cas d'un problème ayant plusieurs solutions, qu'advient-il si l'on ajoute des préférences (par exemple, l'anglais attribue la valeur 5 à la maison bleue et la valeur 2 à la maison rouge, signifiant ainsi qu'il préfère la maison bleue et souhaite y habiter si possible) ?
- Et si l'énigme faisait plutôt intervenir 100 maisons et des milliers d'indices ?

[5] www.solver.com
[6] office.microsoft.com/fr-ca/excel
[7] www.ampl.com
[8] www.gams.com
[9] www.ilog.com/products/oplstudio

Cette dernière question peut sembler farfelue. Il s'agit cependant d'une problématique bien réelle à laquelle doivent faire face de nombreux décideurs. La section suivante présente un survol d'une application nécessitant la résolution d'un problème de taille appréciable.

Application

La création d'un horaire pour une école primaire fait intervenir une multitude de décisions et de contraintes à respecter. Un groupe donné (par exemple le Groupe 1 de l'Année 3) est associé à une personne (l'enseignant ou l'enseignante de ce groupe). Il y a toutefois un certain nombre d'enseignants spécialistes qui enseignent à plusieurs groupes (pour les cours de musique, d'éducation physique et d'anglais, par exemple).

Le problème consiste à créer un horaire valide (ou encore cohérent), c'est-à-dire un horaire qui affecte ces enseignants spécialistes aux divers groupes de l'école en évitant les conflits d'horaire, tout en respectant le nombre d'heures que doit effectuer chacun des spécialistes pour chacun des groupes.

Par exemple, il peut être exigé que l'enseignant d'éducation physique X enseigne 2 heures par 5 jours (sur un horaire de 10 jours) au Groupe 1 de l'Année 3. De plus, on peut désirer que ces deux heures ne soient pas données sur deux journées consécutives. Il peut aussi être souhaité que les horaires soient symétriques, dans le sens qu'un cours donné le jour 4 à 10 :00 soit aussi donné le jour 9 à 10 :00 (afin d'obtenir un horaire stable facile à gérer autant par la direction et le personnel enseignant que par les élèves et leurs parents).

On peut aussi souhaiter que si deux groupes composent l'année 3, alors les cours d'éducation physique de ces deux groupes soient consécutifs (afin de profiter de l'installation des plateaux ; la même chose peut se produire en ce qui concerne la préparation des instruments pour un cours de musique). On pourrait ainsi dire que la combinaison *Enseignant_X-Cours_d'éducation_physique-Année_3-Groupe_1-Jour-Heure* doit être « à côté » de la combinaison *Enseignant_X-Cours_d'éducation_physique-Année_3-Groupe_2-Jour-Heure* (une combinaison vraie ayant la valeur 1 correspondrait alors à une affectation choisie, déterminée par le logiciel d'optimisation).

Vous pouvez sans doute imaginer plusieurs autres contraintes que doit satisfaire un horaire de ce type. Évidemment, il est courant en pratique que l'ensemble des contraintes initialement exigées soit incohérent et mène à la conclusion qu'il n'existe aucun horaire satisfaisant toutes les contraintes. On doit alors « relaxer » le problème en laissant tomber certaines contraintes. Le processus menant à un horaire final demande souvent beaucoup d'effort de ce côté et les habiletés d'un gestionnaire expérimenté favorisent la prise de décision finale. Un outil permettant la création d'un horaire peut grandement faciliter cette lourde tâche et permettre une évaluation rapide de la réalisabilité de certains scénarios (dont l'embauche de personnel spécialisé à temps partiel, par exemple).

L'utilisation de logiciels spécialisés offre des opportunités intéressantes afin de réaliser l'étude de tels problèmes. Des applications peuvent être développées en utilisant ces logiciels et en créant des systèmes intégrés d'aide à la décision qui soient conviviaux. Notons en particulier la force de la librairie de CPLEX ainsi que la facilité d'utilisation de la librairie OptiMax 2000 de MPL. Ces

outils méritent d'être connus et ce, en particulier, par les élèves qui étudient présentement au niveau gradué au sein d'un département de mathématiques appliquées ou de recherche opérationnelle.

Énigmes et jeux mathématiques

Internet propose une quantité pratiquement illimitée d'énigmes et jeux mathématiques de toutes sortes. Des sites tels http ://www.apprendre-en-ligne.net/blog/index.php/Enigmes-casse-tete, http ://www.seed.slb.com/fr/scictr/lab/math/past.htm et http ://www.jeux-maths.com/enigme.htm offrent des énigmes logiques et arithmétiques intéressantes. Des concours peuvent aussi nous lancer des défis non triviaux (par exemple, voir la section *Concours collégiaux de l'Association mathématique du Québec*, au site http ://newton.mat.ulaval.ca/amq/ConcoursCollegiaux.html, ainsi que la section *Compétition de l'Association suisse de recherche opérationnelle*, au site http ://www.asro.ch).

La recherche opérationnelle constitue un champ d'études qui repose sur la résolution de divers problèmes de ce type. Ainsi, tout livre d'introduction à la recherche opérationnelle mentionne de nombreux problèmes faisant intervenir diverses contraintes (des « indices ») ainsi qu'une fonction-objectif à optimiser. Mentionnons le problème du sac à dos, le problème du voyageur de commerce, l'énigme des ponts de Königsberg, la maximisation des profits d'une industrie, les problèmes de mélange, d'affectation, de min-max et plusieurs autres qui demandent des outils de modélisation et des techniques d'optimisation spécialisées. Nul doute que les liens unissant les langages de programmation algébriques et les optimiseurs performants peuvent favoriser la représentation et la résolution de problèmes complexes plus difficiles à étudier autrement.

Conclusion

Une énigme. Une simple énigme qu'une enseignante de 6^{e} année a proposée à ses élèves s'est révélée être un véhicule intéressant afin de présenter un problème de modélisation mathématique ainsi que sa représentation via un langage de programmation algébrique et sa résolution en utilisant un logiciel spécialisé d'optimisation. Il s'agit d'un exemple académique menant à des applications élégantes, complexes et puissantes afin de favoriser la prise de décisions non triviales de façon efficace.