Conception et implémentation d'un outil d'optimisation dans CAPSIS

Partie I : aspects techniques,
Optimisation sous contrainte
Adaptations à une réponse aléatoire

Gilles Le Moguédec, UMR AMAP

Hanitra Rakotoarison, ONF – Département RDI

Mathieu Fortin, UMR LERFoB

Travail effectué dans le cadre du projet :
« Optimisation et viabilité de la gestion forestière en présence de risques »
financement Ecofor et ONF

Optimisation sous Capsis

•Le mode script de Capsis permet de générer **automatiquement** des centaines, voire des milliers, de scénarios de sylviculture et d'en récupérer les résultats.

•Il s'agit donc de faire en sorte de Capsis soit capable de déterminer **lui-même** quelle scénario de sylviculture, appartenant à une famille donnée, permet d'obtenir la **meilleure réponse possible**, évaluée par un certain critère, tout en respectant des contraintes.

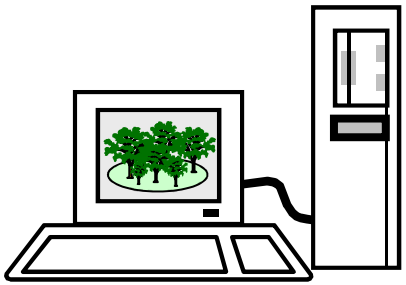
•C'est à l'utilisateur de définir la famille de scénarios, le critère à optimiser et les contraintes à respecter. Au programme de faire le reste.

•Cependant le problème est compliqué par la nature aléatoire de la réponse à un scénario : le même scénario exécuté 2 fois ne donnera pas nécessairement les mêmes résultats, surtout s'il est exécuté dans un environnement incluant des événements aléatoires (contexte de risque).

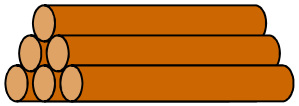
•De plus, l'utilisateur n'est pas nécessairement un spécialiste des techniques d'optimisation, il faut lui faciliter un peu la tâche. *Cf.* présentation de Mathieu Fortin.

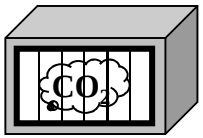
Eléments nécessaires à l'optimisation

- **Un modèle de croissance :**

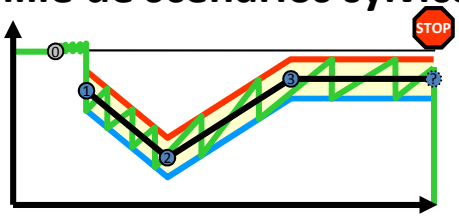


- **Un critère d'évaluation à optimiser (fonction objectif) et éventuellement des performances à atteindre (contraintes)**

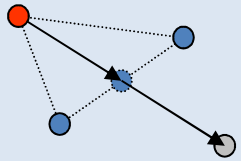


- **Une famille de scénarios sylvicoles (variables de commande)**



- **Une méthode d'optimisation adaptée**



Fonction objectif et contraintes

Pour chaque scénario réalisé, le programme calcule **différents critères de performance** :

- Volume commercial annuel récolté;
- Durée du scénario ;
- Bénéfice net actualisé;
- …

L'utilisateur a la possibilité d'utiliser **tout ou partie** de cette liste (mais seulement cette liste) pour définir la fonction qu'il souhaite **optimiser** et les **contraintes** qu'il veut respecter.

La fonction objectif et les contraintes font intervenir plusieurs des critères élémentaires : il s'agit d'un exemple **d'optimisation multicritère**.

La fonction objectif est une **combinaison linéaire** des critères élémentaires :

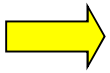
Par exemple : $a \times Volume + b \times Bénéfice$

dont l'utilisateur choisit les valeurs des coefficients a *et* b.

Les contraintes sont des **contraintes à l'inégalité** sur les critères élémentaires :

Par exemple : $Durée \leq 150\ ans$

L'utilisateur doit utiliser au moins un critère pour la fonction objectif, dire s'il veut minimiser ou maximiser l'objectif, mais il n'est pas nécessaire de définir une contrainte.

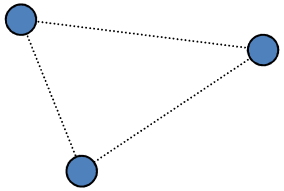


Grande souplesse des problèmes d'optimisation étudiables

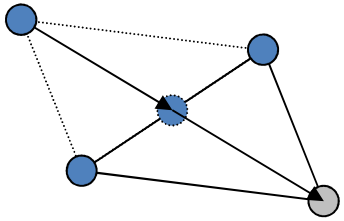
L'algorithme de Nelder & Mead

J.A. Nelder and R. Mead, 1965. *Computer Journal*, vol 7, pp 308-313

Si *p* paramètres à estimer, on génère (*p+1*) points initiaux

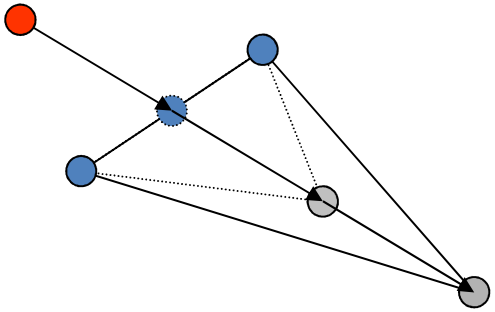


Réflexion du pire de ces points par rapport au centre de gravité des autres

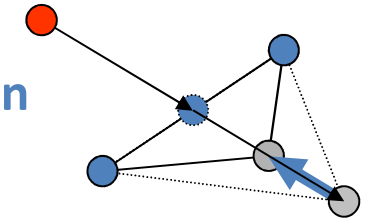


Selon la performance associée à ce point, on le conservera pour l'itération suivante, ou on procèdera à une **extension** ou une **réduction**.

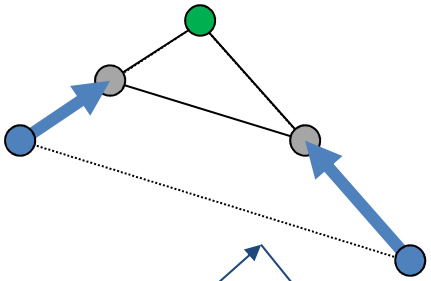
Extension



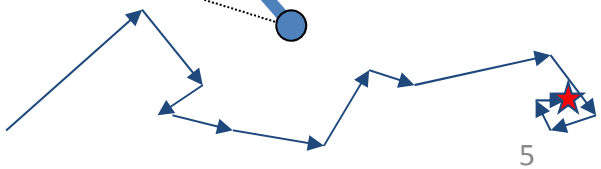
Réduction



Les versions plus récentes de cet algorithme comprennent une étape supplémentaire : la **contraction**.



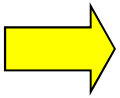
Quelle que soit la version, progression séquentielle



Remarques sur l'algorithme de Nelder & Mead

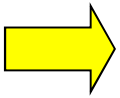
L'algorithme de Nelder-Mead permet d'optimiser une fonction déterministe de $\mathbb{R}^p$ dans $\mathbb{R}$.


Il n'est pas nécessaire que cette fonction soit dérivable, ni même continue.


Adaptable à une grande variété de problèmes.


En contrepartie, l'algorithme de Nelder & Mead est lent.

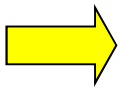

Comme beaucoup d'autres méthodes, il est sensible aux optima locaux.


Nécessité de tester plusieurs points de départ.


Ne permet pas de restreindre l'espace des paramètres à un intervalle;


Ne permet pas d'intégrer des contraintes sur le scénario optimal ;


Ne fonctionne pas sur une réponse aléatoire ;

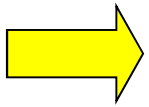

Adaptations nécessaires pour transformer les  en  ou 

Restriction de l'espace des paramètres

Les paramètres sur lesquels travaille l'algorithme de Nelder-Mead prennent leurs valeurs dans $\mathbb{R}$.

Or ceux utilisés pour les scénarios sont souvent bornés.

Par exemple pour le RDI on doit avoir : $0 \leq RDI \leq 1$.



Il suffit de travailler sur une transformation réversible des paramètres.

Exemple : pour le RDI on considère la transformation logit et sa réciproque :

$$]0;1[\longrightarrow \mathbb{R}$$
$$x \mapsto logit(x) = log\left(\frac{x}{1-x}\right)$$

et

$$\mathbb{R} \longrightarrow]0;1[$$
$$u \mapsto logistic(u) = \frac{e^u}{1+e^u}$$

Le programme travaille sur une valeur réelle non bornée qui est ensuite convertie en RDI par la transformation logistique.

Le même type d'astuce peut être utilisé pour convertir un ensemble de *p* valeurs non bornées en un ensemble de *p* valeurs ordonnées, bornées ou non. Utile si certains paramètres de scénarios doivent être ordonnés.

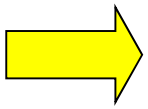
Intégration de contraintes :

Il y a deux types de contraintes : contraintes *a priori* et *a posteriori*

- **Avec une contrainte *a priori*,** on sait avant même de le tester si un scénario respecte la contrainte.

Elle restreint d'emblée la famille de scénarios dans laquelle chercher la solution optimale.

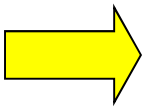
Par exemple, on peut vouloir imposer $RDI \geq \frac{1}{2}$.



Il suffit de restreindre l'espace des paramètres à l'espace admissible.
Cf. diapositive précédente.

- **Avec une contrainte *a posteriori*,** on doit faire tourner le scénario pour constater **après coup** si la contrainte est respectée ou non.

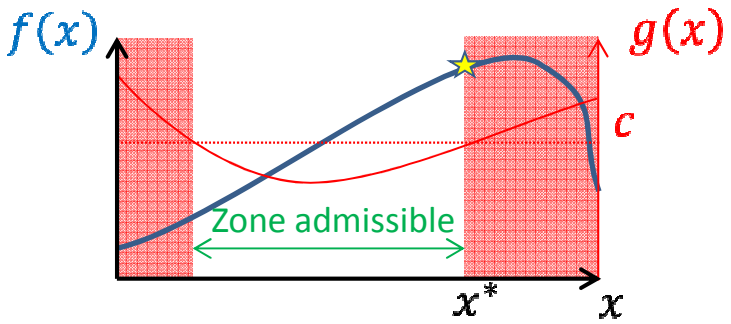
Exemple : on veut maximiser le revenu généré par un scénario sylvicole sous réserve que ce dernier réalise au moins un certain score pour une fonction environnementale (que l'on sait calculer).



Pour un scénario donné, il faut d'abord l'exécuter, pour ensuite constater ses performances financière et environnementale. Ce n'est qu'à ce moment-là qu'on peut dire si le scénario est admissible ou non.

Intégration de contraintes *a posteriori*

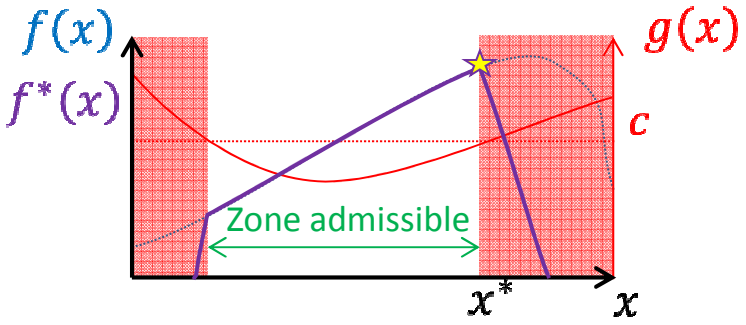
Soient f et g deux fonctions d'un paramètre de scénario x. On cherche la valeur de x permettant de maximiser f sous la contrainte $g(x) \leq c$.



On construit une fonction de pénalité $p(x)$ mesurant l'écart à la contrainte :

$$p(x) = \begin{cases} 0 & si\ g(x) \leq c \\ M \times [g(x) - c] & si\ g(x) > c \end{cases}$$

Et on maximise la fonction $f^*(x) = f(x) - p(x)$



La constante $M > 0$ doit être grande mais pas trop :

- Assez grande pour fortement pénaliser les scénarios non admissibles;
- Assez petite pour donner au programme une direction vers laquelle chercher la zone admissible.

Adaptation à une réponse aléatoire

On veut optimiser la réponse f à un scénario sylvicole défini par le paramètre x.

Mais cette réponse est **aléatoire** :

- Génération du peuplement initial;
- Lors d'une éclaircie, on définit un taux de coupe par classe de diamètre mais pas quels arbres seront éclaircis;
- Des événements aléatoires peuvent survenir (tempêtes, fluctuations de prix, …);
- …

⇒ Quand on applique le scénario x, on n'observe pas la réponse $f(x)$ mais une variable aléatoire $F(x)$.

⇒ Cela n'a plus de sens de vouloir optimiser $f(x)$. A la place on va chercher à optimiser l'**espérance** $\mathbb{E}[F(x)]$.

On **approche** cette espérance par la **moyenne** $\bar{F}(x) = \frac{1}{n}\sum_{i=1}^{n} F_i(x)$, où les $\{F_i(x)\}_{i=1..n}$ sont les réponses obtenues sur des réalisations indépendantes du scénario x.

Plus n est grand, plus précise est cette estimation. On se fixe un **objectif de précision** $\sigma_{\bar{F}}$, basé sur **l'écart-type empirique** de $\bar{F}(x)$ (pas de $F_i(x)$), et on effectue autant de simulations que nécessaire pour atteindre cet objectif.

Cependant, pour éviter des temps de calcul trop importants, on fixe une limite n_{max} à ne pas dépasser, même si l'objectif de précision n'est pas atteint.

⇒ Tests **préalables** nécessaires pour donner des valeurs **raisonnables** à $\sigma_{\bar{F}}$ et n_{max}.

Exploration de l'espace des paramètres

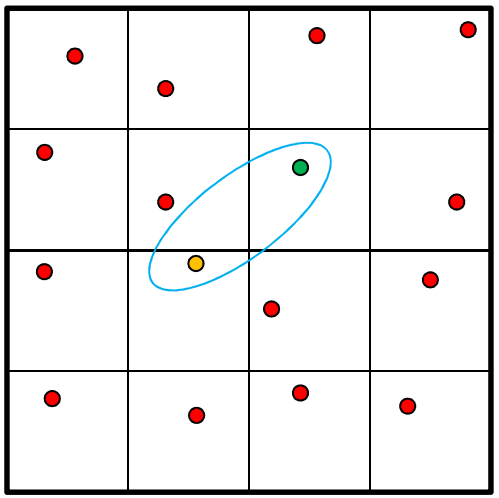
- D'une part on n'est jamais à l'abri de tomber sur un optimum local;
- D'autre part il peut être difficile de trouver un scénario initial respectant les contraintes;

⇨ Avant l'optimisation proprement dite, on effectue une exploration préalable de l'espace des paramètres

Pour cela, on se donne un paramètre $k \in \mathbb{N}$:

- $k = 0$: on part d'un point fixe préalablement défini et on lance directement l'optimisation ;
- $k = 1$: on part d'un point aléatoire et on lance directement l'optimisation ;
- $k \geq 2$: on génère une grille de recherche basée sur le découpage de l'espace de chaque paramètre du scénario en k parties, et on tire aléatoirement un point à l'intérieur de chacune des mailles de la grille.
- On évalue le scénario en chacun de ces points et on garde les deux meilleurs pour l'initialisation de l'optimisation.

Exemple pour 2 paramètres et $k = 4$:



- *Ne pas prendre k trop grand : les scénarios actuels sont définis par 6 paramètres. Il faut donc évaluer préalablement k^6 scénarios avant de commencer l'optimisation.*
- *Pour la phase exploratoire, on a juste besoin d'un ordre de grandeur de la performance. On prend donc n_{max} petit et on l'augmente pour la phase d'optimisation.*

Conclusion : les paramètres techniques

Il est possible de réalisation une optimisation de sylviculture sous Capsis, mais au prix de nombreux réglages techniques :

Paramétrage du problème d'optimisation :

- Coefficients de la combinaison linéaire à optimiser ;
- Bornes des contraintes éventuelles, coefficient de pénalité M.

Paramétrage général de l'algorithme de Nelder-Mead :

- Nombre max d'itérations autorisées ;
- Nombre max d'évaluations de la fonction autorisées ;
- Critère de convergence absolue
- Critère de convergence relative

Adapter la valeur à la variabilité de la fonction !

Paramétrage des adaptations

- Critère de précision $\sigma_{\bar{F}}$ pour la réponse moyenne
- Nombre n_{max} de répétitions autorisées pour chaque scénario
- Pas k de la grille d'exploration

- Phase exploratoire
- Phase d'optimisation

➡ Le programme fonctionne bien, mais il nécessite un apprentissage

(sans parler de l'interprétation des résultats)

Mathieu, c’est l’heure de la démo !