

NRC Publications Archive Archives des publications du CNRC

Algorithme d'apprentissage pour inférer les paramètres de proaftn Benabbou, L.; Guitouni, A.; Belacel, N.

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. /
La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.

Publisher's version / Version de l'éditeur:

*Administrative Sciences Association of Canada Conference 2004 (ASAC 2004)
[Proceedings], 2004*

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
<https://nrc-publications.canada.ca/eng/view/object/?id=4714c65a-9bfc-4a53-9055-7f14ec39cb0e>
<https://publications-cnrc.canada.ca/fra/voir/objet/?id=4714c65a-9bfc-4a53-9055-7f14ec39cb0e>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at
<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site
<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at
PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.



National Research
Council Canada

Institute for
Information Technology

Conseil national
de recherches Canada

Institut de technologie
de l'information

NRC - CNRC

Algorithme d'apprentissage pour inférer les paramètres de PROAFTN *

Benabbou, L., Guitouni, A., et Belacel, N.
Juin 2004

* publié dans les Actes de la Conférence de l'Association des Sciences
Administratives du Canada 2004 (ASAC 2004). Québec, Québec, Canada.
Du 5 au 8 juin 2004. NRC 47128.

Copyright 2004 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report,
provided that the source of such material is fully acknowledged.

ASAC 2004
Québec, QUÉBEC

Loubna Benabbou
Adel Guitouni
Faculté des sciences de l'administration
Université Laval
Nabil Belacel
National Research Council
Institute for Information Technology-e-Business, Saint John, Canada

ALGORITHME D'APPRENTISSAGE POUR INFÉRER LES PARAMÈTRES DE PROAFTN

PROAFTN est une procédure de classification multicritère qui permet d'affecter directement les actions potentielles aux différentes classes. Dans ce travail nous proposons un algorithme d'apprentissage qui permettra d'ajuster automatiquement les paramètres de cette méthode. Cet algorithme fait appel aux outils de recherche locale et de méta-heuristique pour minimiser l'écart entre les résultats de classification obtenus par la méthode PROAFTN et ceux donnés à priori dans l'ensemble d'apprentissage.

1 Revue des problèmes de la classification multicritère

La classification est utilisée naturellement depuis déjà bien longtemps pour comprendre et communiquer notre vision du monde (par exemple les espèces animales, minérales ou végétales). La classification consiste à affecter des actions (objets, individus ou autres), à des catégories ou à des classes plus au moins prédéfinies. Les problèmes de classification ont suscité l'intérêt dans plusieurs domaines de recherche. Notamment en aide multicritère à la décision, en statistique, en datamining et en intelligence artificielle. Dans le domaine d'aide multicritère à la décision, Moscarola et Roy (1977) ont été, à notre connaissance, les premiers à introduire une méthode de classification multicritère; on parle ici beaucoup plus de tri que de classification. Le principe général de ces méthodes est basé sur les préférences du décideur, tout en considérant des catégories ordonnées.

Le champ d'application de la classification est très large. Elle traite une multitude de problèmes d'ordre intellectuel, économique ou commercial. Dans la littérature, nous trouvons plusieurs problèmes qui s'inscrivent dans l'optique de classification multicritère (Henriet, 2000) : l'évaluation des dossiers de crédits (J. Moscarola et B. Roy, 1977), la reconnaissance de la parole (S.K. Pal et D.D. Majumder, 1977), la reconnaissance des cafés (L. Foulloy et al, 1996), l'évaluation environnementale (C.Arondel et P. Girardin, 1998) et le diagnostic médical (N. Belacel, 2000, 1999). Cette liste exhaustive de certaines applications des méthodes de classification peut être généralisée vers d'autres domaines. La section suivante est dédiée à la présentation de la méthode PROAFTN, qui va faire l'objet d'amélioration dans notre travail, comme l'une des plus récentes méthodes de la classification multicritère.

2 PROAFTN : PROCÉDURE d'Affectation Floue dans le cadre de la problématique du Tri Nominal

PROAFTN fait partie des méthodes de classification supervisée. Elle détermine la classe d'affectation d'une action à partir des relations de ressemblances floues déterminées par les indices d'indifférence (Belacel, 1999, 2000). La procédure PROAFTN se caractérise par une affectation graduelle ou des actions aux différentes classes. Cette méthode peut combiner les deux types d'apprentissage : déductif et inductif, ce qui n'est pas le cas des autres méthodes. Comme son nom l'indique, cette méthode traite la problématique du tri nominal. PROAFTN traite les problèmes avec des données de nature qualitative ou quantitative.

À chaque prototype b_i^h pour chaque classe h , et pour chaque critère g_j on associe l'intervalle d'évaluation : $[S_j^1(b_i^h), S_j^2(b_i^h)]$. Avec $S_j^2(b_i^h) \geq S_j^1(b_i^h)$, pour $j = 1 \dots n$, $h = 1 \dots K$ et $i = 1 \dots L_h$. Ces intervalles d'évaluation peuvent être considérés comme des intervalles pessimistes. Les seuils de discrimination $q_j^-(b_i^h)$ et $q_j^+(b_i^h)$ (avec $q_j^-(b_i^h), q_j^+(b_i^h) \geq 0$), ont été introduits afin de tenir compte de l'indifférence faible d'une action par rapport à un prototype. Dans un premier temps, PROAFTN détermine l'indice de concordance $c_j(a, b_i^h)$ et de discordance $D_j(a, b_i^h)$ de l'action a par rapport au prototype b_i^h . L'indice de concordance partiel $c_j(a, b_i^h)$ est obtenu par la courbe d'indifférence suivante :

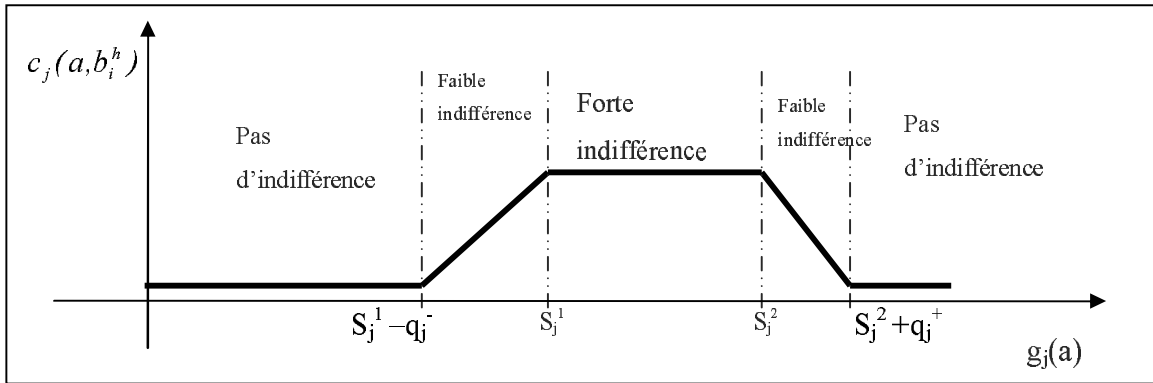


Figure 1 : Représentation de l'indice de concordance

Pour déterminer l'indice de discordance, on intègre les seuils de veto $v_j^+(b_i^h)$ et $v_j^-(b_i^h)$ tel que $v_j^+(b_i^h) \geq q_j^+(b_i^h)$ et $v_j^-(b_i^h) \geq q_j^-(b_i^h)$. Avec ces seuils nous pouvons construire la courbe de discordance suivante :

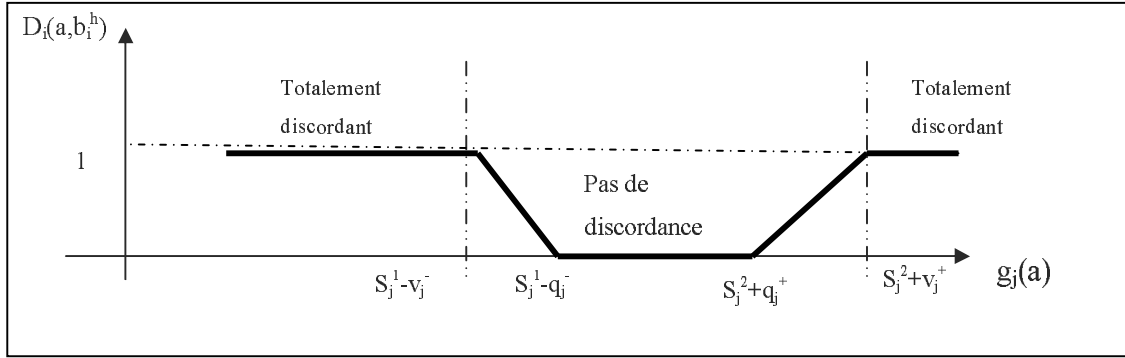


Figure 2 : Représentation de l'indice de discordance

En appliquant le principe de concordance et de non-discordance, nous déterminons l'indice d'indifférence de l'action a par rapport au prototype i de la classe h : b_i^h .

$$I(a, b_i^h) = \sum_j w_j^h c_j(a, b_i^h) * \prod_j (1 - D_j(a, b_i^h))^{w_j^h} \quad (1)$$

En calculant cet indice sur tous les prototypes de la classe h , nous pouvons déterminer l'indice d'appartenance global, de l'action a par rapport à la classe h tel que :

$$d(a, C^h) = \max\{I(a, b_1^h), \dots, I(a, b_{L_h}^h)\} \quad (2)$$

La décision d'affectation de l'action a à la classe h s'annonce comme suit :

$$a \in C^h \quad \text{Sietsi} \quad d(a, C^h) = \max_k \{d(a, C^k)\} \quad \text{avec } k = 1 \dots K \quad (3)$$

L'analyse de la méthode PROAFTN nous permet de constater l'existence d'une procédure d'affectation multicritère floue, avec un nombre illimité de catégories. Elle combine les deux techniques d'apprentissage inductif et déductif, tout en assurant une interaction avec le décideur afin de déterminer les paramètres. Cette interaction est très importante dans un processus de décision. Seulement, le nombre de paramètres que doit déterminer le décideur augmente d'une manière très rapide avec l'augmentation du nombre des classes et du nombre des critères. Il serait opportun de déterminer les paramètres et les faire valider par la suite par le décideur. En effet, pour 4 classes et 4 critères, le décideur doit déterminer 64 seuils, plus les poids des critères, ceci en supposant qu'on a un seul prototype par classe. Alors que pour 10 classes et 10 critères le chiffre atteint 1000 paramètres à déterminer !!.

Sur le plan pratique, le décideur a bel apprécié l'interaction et l'intervention dans le processus de décision Mais déterminer des centaines de paramètres demeure une tâche très difficile, éliminant ainsi sa coopération et voir même son intérêt d'intervention dans notre processus de décision. Sur le plan théorique, déterminer des centaines de paramètres peut donner naissance à des erreurs de jugement et d'estimation, entravant ainsi l'application de la méthode PROAFTN, et par la suite peut aboutir à des résultats non satisfaisants.

Compte tenu de l'enjeu de l'intervention du décideur dans le processus de décision, de la difficulté d'estimer ce nombre important de paramètres, des résultats insatisfaisants que peuvent engendrer des erreurs d'estimation, de l'essor que connaît l'application de la méthode PROAFTN dans plusieurs domaines plus particulièrement le domaine médical (Belacel et Boulassel, 2001 et 2004; Larburu et al, 2003) et de l'existence de plusieurs techniques d'optimisation et de classification, nous jugeons qu'une recherche d'une démarche qui permettra d'ajuster automatiquement la détermination des paramètres de PROAFTN est tout à fait justifiée. Dans un premier temps, la démarche à développer déterminera les paramètres de PROAFTN, ces paramètres vont être validés par la suite par le décideur. Si jamais ces paramètres ne correspondent pas à ses préférences, l'algorithme permettra de les ajuster automatiquement.

Pour présenter la démarche développée, nous avons organisé ce document comme suit : Nous allons d'abord proposer la modélisation mathématique du problème de détermination des paramètres de la méthode PROAFTN. Par la suite, nous allons présenter une revue des techniques d'apprentissage pour la résolution des problèmes de classification et des techniques de local search. Ceci va nous permettre de ressortir l'approche pour déterminer automatiquement ces paramètres. Cette approche va nous guider pour concevoir l'algorithme qui va résoudre le problème formulé. À la fin, nous allons analyser la performance de cet algorithme en l'appliquant sur un cas pratique.

3 Modélisation du problème de détermination des paramètres de PROAFTN

Une des options d'améliorer la méthode PROAFTN est de déterminer directement les paramètres en minimisant les erreurs d'affectation à partir d'un ensemble d'apprentissage. Ce même principe a été utilisé par Mousseau et al (2001) pour inférer les paramètres de la méthode Electre tri. Après l'application de la méthode PROAFTN, on trouve les degrés d'affectation de chaque action a_i ($i = 1 \dots m$) de l'ensemble d'apprentissage par rapport à une classe C^h ($d_{ih}(a_i, C^h)$). Comme les classes des actions de l'ensemble d'apprentissage sont données a priori, alors on a automatiquement l'indice de décision d'affectation α_{ih} , tel que $\alpha_{ih} \in \{0,1\}$ avec $\alpha_{ih}=1$ si $a_i \in C^h$, $\alpha_{ih}=0$ sinon. La fonction objectif du programme minimise la distance entre $d(a_i, C^h)$ et α_{ih} , sous les contraintes sur les paramètres utilisés par la méthode PROAFTN. Ainsi, le programme mathématique (P), qui nous permet de déterminer les paramètres S^1, S^2, q^+, q^-, W de PROAFTN peut être formulé comme suit (Belacel *et al.*, 2001) :

$$\begin{cases} \min \sum_{i=1}^m \sum_{h=1}^k (d_{ih}(S^1, S^2, q^+, q^-, W) - \alpha_{ih})^2 \\ S_{jh}^1 - q_{jh}^- \leq S_{jh}^2 + q_{jh}^+ \\ 0 \leq w_{jh} \leq 1 \\ \sum_{j=1}^n w_{jh} = 1 \\ q_{jh}^-, q_{jh}^+ \geq 0 \\ j : 1 \dots n, \quad h : 1 \dots k \end{cases} \quad (4)$$

La fonction objectif du programme mathématique (4) exprime l'erreur de classification. L'erreur de classification en apprentissage peut prendre plusieurs formes. On a retenu ici la fonction de perte quadratique entre $d(a_i, C^h)$ et α_{ih} . Ainsi le problème (4) minimise l'erreur quadratique de la classification sous les contraintes sur les paramètres de PROAFTN. Nos variables de décision sont les poids W_{ih} , les seuils S_1, S_2, q^+ et q^- . Le programme proposé peut faire l'objet de plusieurs simplifications. Pour simplifier le problème, nous supposons que les poids des critères sont déjà déterminés par le décideur ou en interaction avec lui; en appliquant AHP par exemple et que le nombre de prototypes, se limite à un seul prototype par classe ($L_h=1$). Pour l'instant, nous allons considérer les variables de décision S^1, S^2, q^+ et q^- , donc le nouveau programme va s'écrire :

$$\begin{cases} \min \sum_{i=1}^m \sum_{h=1}^k (d_{ih}(S_1, S_2, q^+, q^-) - \alpha_{ih})^2 \\ S_{jh}^1 - q_{jh}^- \leq S_{jh}^2 + q_{jh}^+ \\ q_{jh}^-, q_{jh}^+ \geq 0 \\ j : 1....n, \quad h : 1....k \end{cases} \quad (5)$$

Le programme mathématique consiste à minimiser les erreurs d'affectation, i.e., minimiser la différence quadratique entre le degré d'affectation $d(a_i, C_h)$ obtenu par PROAFTN et la valeur de l'indice de décision d'affectation donnée a priori α_{ih} . Par exemple, si l'indice $\alpha_{ih} = 1$ alors le degré $d(a_i, C_h)$ devrait converger vers la valeur 1. La fonction objectif est une fonction non-linéaire et elle n'est ni convexe ni concave et en plus elle n'est pas différentiable. Ainsi, on ne peut pas résoudre le programme mathématique (5) par les méthodes classiques de la programmation non-linéaire, comme par exemple les méthodes de gradient (réduit, généralisé,...), les méthodes de descentes ou par les méthodes de points intérieurs. Par conséquent, il serait intéressant de développer des heuristiques ou des méta-heuristiques pour résoudre le programme non-linéaire (5).

Nous nous sommes placés dès le départ dans un contexte de classification avec apprentissage supervisé. Donc nous avons un échantillon des données classées au préalable. Cet échantillon va être divisé en trois ensembles : ensemble d'apprentissage, ensemble de test et ensemble d'évaluation. Ces trois ensembles devront être distincts (n'avoir aucun enregistrement en commun). L'ensemble d'apprentissage va être utilisé pour résoudre le problème (5) et pour construire le modèle initial. C'est depuis cet ensemble que le système va calculer les différents paramètres de PROAFTN. Une fois les paramètres calculés, il faut vérifier comment ils se comportent sur l'ensemble de test. Celui-ci va permettre d'ajuster les valeurs trouvées à l'étape précédente et les rendre moins sensibles à l'ensemble d'apprentissage. Enfin, les paramètres seront testés sur l'ensemble d'évaluation. Si les résultats obtenus sont proches des attentes du décideur, on pourra alors valider le système. Dans le cas contraire, grâce à l'algorithme on pourra ajuster automatiquement les paramètres. Cette étape d'évaluation permet de maintenir l'aspect interactif avec le décideur et d'intégrer ses préférences dans l'affectation des actions.

4 Revue des techniques d'apprentissage pour la classification

Les techniques d'apprentissage pour la classification peuvent être regroupées en deux catégories selon le type d'apprentissage supervisé ou non (Belacel, 1999):

- Apprentissage non supervisé : on retrouve dans cette catégorie les méthodes de classification automatiques ou clustering. Ces méthodes peuvent être hiérarchiques (ascendantes ou descendantes) ou non hiérarchique (partitionnement)
- Apprentissage supervisé : on parle ici de méthodes d'affectation. Ces méthodes peuvent être divisées à leur tour en trois groupes selon le type de raisonnement de l'apprentissage : les méthodes d'apprentissage inductif (méthodes statistiques et machine learning), les méthodes d'apprentissage déductif (systèmes experts) et les méthodes d'affectation multicritère.

Les techniques d'apprentissage permettent de résoudre un problème de prise de décision très complexe en se basant sur des exemples passés. Elles permettent d'établir des règles à partir des données disponibles ou des expériences réussies a priori. Elles font appel à des méthodes mathématiques et statistiques qui permettent d'exploiter le plus grand nombre de possibilités, tout en exploitant la capacité de calcul de l'ordinateur sans avoir besoin d'une interaction directe avec le décideur.

Les techniques d'apprentissage reposent sur les schémas d'inférence. On distingue deux types de raisonnement : La déduction et l'induction. La déduction est le type de raisonnement le plus utilisé et le plus familier. Son atout majeur est qu'il ne laisse pas de place au doute. Quant au raisonnement inductif, il consiste à tirer des conclusions à partir d'une série de faits. La certitude n'est pas absolue et sera donc associée à une probabilité. Plus les faits appuyant l'hypothèse sont nombreux, plus la probabilité que la conclusion soit exacte est forte.

Malgré la diversité des techniques d'apprentissage pour la classification, aucune méthode, à notre avis, ne propose une procédure claire de détermination des différents paramètres. Toutefois, parcourir ces méthodes nous a permis de ressortir la pertinence d'utiliser l'apprentissage dans notre méthode de résolution. Aussi, elle nous a montré la nécessité de développer une nouvelle approche pour déterminer automatiquement les paramètres. Notre programme mathématique doit déterminer les nouveaux paramètres à partir des résultats de classification des données au passé. Donc, l'algorithme que nous devons développer, doit construire un modèle général à partir de la classification des données a priori. Ceci nous mène à se placer dans le cadre d'un apprentissage inductif supervisé.

5 Revue des techniques de recherche locale

Local search ou la recherche locale consiste à partir d'une solution initiale et de chercher dans son voisinage, de manière continu, une solution meilleure. Cette technique a été développée pour résoudre des problèmes combinatoires, qui sont en général *NP-Dur*, en essayant de trouver la solution optimale dans un espace fini ou un espace infini et dénombrable de solutions réalisables. Ce domaine de recherche en mathématique discrète a été largement exploité ces

dernières années. Aarts et al (1997) ont cité les travaux de Lawler (1976) de Papadimitriou et al. (1982), de Schrijver (1986) et de Nemhauser et al. (1988) comme de très bonnes références pour les néophytes dans le domaine. Pour une revue bibliographique plus riche, ils ont fait référence aux livres d'O'Heigartaigh, Lenstra et Ronnooy Kan (1985) et Dell'Amico, Maffioli et Martello (1997).

L'une des plus simples fonctions de voisinage, est celle qui est basée sur k échanges des voisins. On a un ensemble de voisins, on échange k fois ces voisins jusqu'à tomber sur un optimum local à chaque fois. Pour des valeurs petites de k , le voisinage est très facilement exploré, mais la qualité des solutions est à revoir. Pour un k plus grand on a des chances pour trouver des solutions meilleures, mais le temps de calcul est très important. D'autres méthodes, basées toujours sur la recherche de voisinage, sont beaucoup plus élaborées et donnent de meilleurs résultats mais avec une complexité supérieure. Telles que : le recuit simulé, la recherche taboue, les algorithmes génétiques et les réseaux de neurones.

Plus récemment, une nouvelle technique de recherche dans le voisinage a vu le jour : Variable Neighborhood Search (VNS). C'est une métaheuristique pour la résolution de problèmes d'optimisation combinatoire et globale. L'idée de base de VNS est de changer systématiquement le voisinage au sein d'une recherche locale. Il existe plusieurs variétés de cette méthode, nous allons présenter VNS inspirée des travaux de P.Hansen et N. Mladenovic, (2001). Contrairement aux autres métaheuristiques qui suivent une trajectoire bien définie, VNS explore tout le voisinage même à des distances importantes de la solution en cours. Cet algorithme (saute) aléatoirement d'une solution à une nouvelle, si une amélioration a été constatée. Dans ce contexte, certaines variables de la solution vont rester à leur optimum local, et peuvent être utilisées pour trouver des résultats beaucoup plus intéressants (Figure 3). Pour réussir cette recherche systématique et pour construire les différentes structures des voisins, il faut définir une métrique qui nous permet de calculer la distance entre les solutions. Cette méthode a fait ses preuves pour résoudre plusieurs problèmes : le voyageur de commerce, théorie de la location discrète, problème p -median, programmation mathématique dans un contexte continu, le minimum du carré d'une distance, programme de programmation bilinéaire, avec contraintes bilinéaires.

Les techniques de la recherche locale ont fait leur preuve dans plusieurs domaines de recherche. Toutefois, pour résoudre notre programme mathématique, nous devons trouver une technique simple qui fait appel à l'apprentissage et qui est adaptée pour notre espace de solutions réalisables. Nous avons besoin d'une technique qui permet de résoudre des problèmes d'optimisation combinatoire, tout en explorant au maximum l'espace de solutions réalisables. Nous jugeons que la méthode VNS réalise ces objectifs.

L'algorithme d'apprentissage que nous avons développé pour résoudre le problème de détermination des paramètres de PROAFTN est basé sur la métaheuristique VNS. L'algorithme va commencer par construire un ensemble de voisins, dans l'espace des solutions réalisables, et à chercher le point qui minimise la fonction objectif en se basant sur une heuristique de local search et changer de voisinage jusqu'à la vérification de la condition d'arrêt. Cet algorithme est représenté dans la figure (3).

Initialisation :

- Choisir N_k ($k=1, \dots, k_{\max}$) l'ensemble des voisins et le paramètre $k_{\max}$,
- Trouver une solution initiale x ,
- Choisir une condition d'arrêt.

Répéter les étapes suivantes jusqu'à ce que la condition d'arrêt soit vérifiée :

1) $k \leftarrow 1$,

2) Répéter ce qui suit jusqu'à $k=k_{\max}$.

a) Exploration du voisinage :

- Générer aléatoirement un point x' du $k^{\text{ième}}$ voisinage de x ($x' \in N_k(x)$)

b) Local search:

- Appliquer une méthode adaptée pour trouver le meilleur voisin (le minimum local) x'' du voisinage N^k de x'

c) Bouger ou non :

- Si la solution x'' est meilleure que x alors $x \leftarrow x''$ et aller à l'étape (1), Sinon $k \leftarrow k+1$ et aller à l'étape (2).

Figure 3 : Algorithme VNS

6 Algorithme d'apprentissage pour inférer les paramètres de PROAFTN

6.1 Détermination des paramètres initiaux

Avant de lancer l'algorithme, il faut déterminer trois éléments essentiels: le voisinage, la solution initiale, la condition d'arrêt. La détermination du voisinage dépend notamment du problème à résoudre, la nature de la fonction objectif et l'espace des solutions réalisables. La solution initiale peut être choisie au hasard ou encore de façon avisée, son choix va directement influencer la solution finale ou la qualité de celle-ci. La condition d'arrêt, est la disposition par laquelle l'algorithme va terminer la recherche de la solution. Une condition d'arrêt trop grande va nous coûter un temps de recherche qui est inutile et une condition d'arrêt trop courte va nous empêcher d'arriver à la meilleure solution.

La méthode d'élaboration du voisinage consiste à définir le voisinage en utilisant une distance euclidienne très simple : le voisinage $N(x^k)$ du point $x^k \in A$, (A l'ensemble des solutions réalisables) à l'itération k est défini par :

$$N(x^k) = \{x \in A / 0 < \|x - x^k\| \leq \alpha_k\} \quad (6)$$

Avec $k : 1 \dots k_{\max}$ et $\alpha_k > 0$ dépend de l'itération k . Le paramètre $k_{\max}$ peut être déterminé Par : $k_{\max} = \min\{10, C\}$ avec C le nombre de classes.

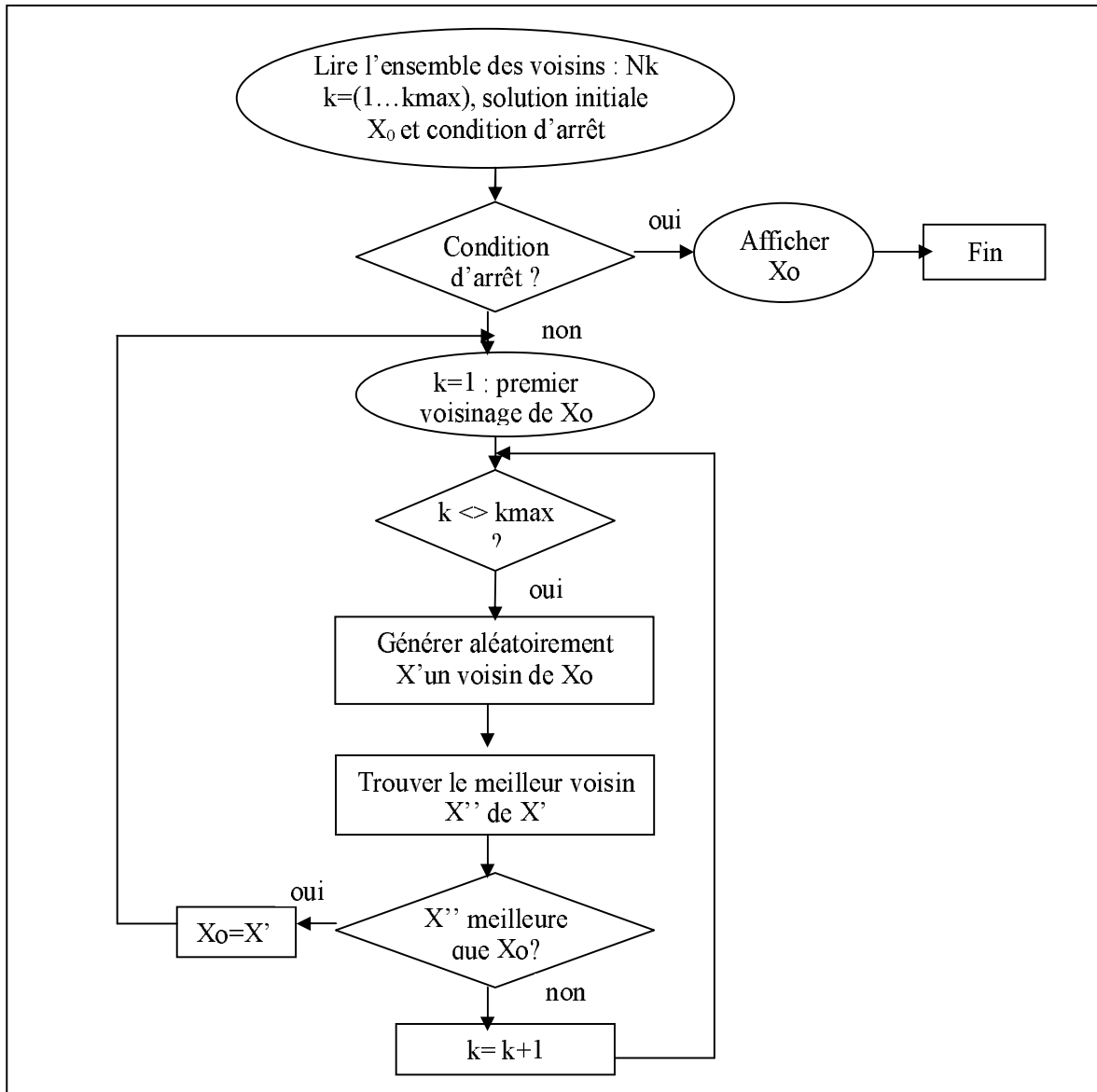


Figure 4 : Approche générale de l'algorithme d'apprentissage pour la détermination des paramètres de PROAFTN

Pour α_k nous avons retenu la définition suivante :

$$\alpha_k = \frac{Max - Min}{k} \quad (7)$$

Avec Max et Min sont le maximum et le minimum des évaluations des actions de l'ensemble d'apprentissage sur un critère donné.

La solution initiale X_0 est la matrice constituée des vecteurs seuils initiaux S^1_0, S^2_0, q^+_0 et q^-_0 . Ces vecteurs seuils initiaux vont être déterminés comme suit :

- Pour S^1_0 : pour chaque critère et pour chaque classe, le minimum des évaluations de toutes les actions de l'ensemble A appartenant à la classe en question.
- Pour S^2_0 : pour chaque critère et pour chaque classe, le maximum des évaluations de toutes les actions de l'ensemble A appartenant à la classe en question.
- À partir de S^1_0 et S^2_0 , on peut définir par une méthode statistique q^+_0 et q^-_0 .

q^+ et q^- sont des seuils qui dépendent de $S1$ et $S2$, ces deux seuils peuvent être déterminés statistiquement en utilisant l'écart type de la distribution de l'ensemble d'apprentissage.

$$\bullet \quad q^+ = S2 + \alpha\sigma \quad (8)$$

$$\bullet \quad q^- = S1 - \alpha\sigma \text{ (avec } \alpha > 0) \quad (9)$$

Avec σ l'écart type de la distribution des évaluations de l'ensemble d'apprentissage sur un seul critère. Pour calculer σ , nous allons prendre les évaluations des actions de l'ensemble d'apprentissage, qui va représenter notre échantillon avec une cardinalité N. Puis calculer la fréquence (f) des valeurs des évaluations et donner une estimation de probabilité : $P=f/N$. Puisque nous travaillons avec des évaluations qui sont des valeurs discrètes, alors nous avons :

$$\begin{aligned} \bullet \quad E(X) &= \sum P \cdot X \\ \bullet \quad Var(X) &= E(X^2) - (E(X))^2 \\ \bullet \quad \sigma(X) &= \sqrt{Var(X)} \end{aligned} \quad (10)$$

Nous allons garder le même σ pour tous les prototypes par un seul critère. Quant à α nous pouvons le fixer en tenant compte du maximum et du minimum des évaluations selon un critère donné et des valeurs de $S1$ et $S2$; nous proposons la formulation suivante :

$$\begin{aligned} \bullet \quad &\text{Pour la solution initiale } \alpha = 1, \\ \bullet \quad &\text{Pour les autres : } \alpha = \lceil \text{Min}\{((\text{max}-S2)/\sigma); ((S1-\text{min})/\sigma)\} \rceil \end{aligned} \quad (11)$$

Comme expliqué précédemment, le choix de la condition d'arrêt doit être réalisé d'une manière judicieuse pour ne pas avoir un temps de recherche ni trop long ni trop court. Nous avons choisi de fixer le nombre maximal des itérations.

6.2 Méthode de détermination du minimum local

Le choix de la méthode de détermination du minimum local dépend essentiellement de la nature de l'espace des solutions réalisables. Nous allons dans un premier temps essayer de déterminer cet espace, par la suite, en appliquant la méthode statistique proposée dans la détermination des paramètres, nous pouvons déterminer q^+ et q^- à partir de S_1 et S_2 , donc notre variable de décision X va se réduire à X (S_1, S_2). Aussi si on veut déterminer l'espace des solutions réalisables, alors on a pour chaque critère et pour chaque classe :

$$\begin{aligned} - \quad &S_1 < S_2, \\ - \quad &Min \leq S_1 < Max \\ - \quad &Min < S_2 \leq Max \end{aligned} \quad (12)$$

Avec le Min est le minimum des évaluations de l'ensemble des actions d'apprentissage par critère et par classe, et le Max est le maximum des évaluations de l'ensemble des actions d'apprentissage par critère et par classe. Si nous voulons représenter graphiquement l'espace des solutions réalisables A, alors nous avons l'espace représenté dans la figure (5)

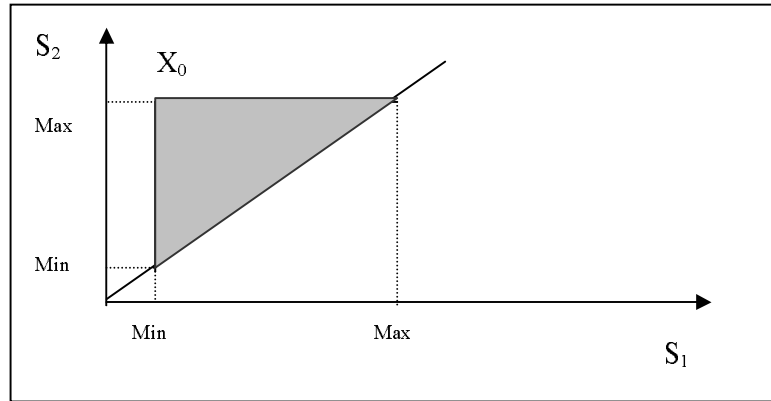


Figure 5 : Espace des solutions réalisables

En appliquant la méthode VNS, pour chaque voisinage N_k à l'itération k , nous allons trouver les voisins X_k . La nature convexe de l'espace des solutions réalisables nous a poussé à réaliser la recherche du minimum local par segment, ce qui nous a mené à appliquer le Golden Section Algorithm (GSA) comme la méthode de recherche local dans VNS.

Initialisation :

- Fixer la constante α (α est le nombre d'or = $(\sqrt{5}-1)/2$)
- Choisir t la largeur finale de l'intervalle d'incertitude (la tolérance)
- Choisir maxits notre condition d'arrêt (Nombre maximal d'itérations)
- Soit $[a_1, b_1]=[a,b]$ l'intervalle initial d'incertitude

Pour $k := 1$ jusqu'à maxits faire

Étape 1:

- Si $b_k - a_k < t$ alors stop
- Sinon, considérer x_1 et x_2 définis ci dessous et aller à l'étape 2
 - $x_1 = a_k + \alpha(b_k - a_k)$
 - $x_2 = b_k - \alpha(b_k - a_k)$

Étape 2:

- Si $f(x_1) < f(x_2)$
 - Alors $a_{k+1} \leftarrow a_k$ et $b_{k+1} \leftarrow x_2$
 - Sinon $a_{k+1} \leftarrow x_1$ et $b_{k+1} \leftarrow b_k$
- $k \leftarrow k+1$ et aller à l'étape 1.

Fin pour.

Figure 6 : Golden Section Algorithm

Golden Section Algorithm (GSA). Le GSA est un algorithme itératif qui vise à réduire un intervalle d'incertitude jusqu'à arriver à une tolérance fixée au préalable, afin de trouver le minimum local en utilisant le nombre d'or (figure 7). Cet algorithme s'applique sur un segment $[a, b]$ tel que $f(a) < f(b)$, ce segment va représenter notre intervalle d'incertitude initial, l'algorithme vise à réduire à chaque itération cet intervalle, en utilisant le nombre d'or $(\sqrt{5}-1)/2$, jusqu'à arriver au plus petit intervalle avec une largeur t qui est notre tolérance de différence entre les deux limites du segment ce qui représente notre minimum local de la fonction f (Chase, 2001).

6.3 Algorithme d'apprentissage pour inférer des paramètres de PROAFTN

L'algorithme d'apprentissage pour inférer des paramètres de PROAFTN est constitué de quatre étapes (Figure 7). La première étape d'initialisation consiste à déterminer les différents paramètres et constantes de l'algorithme et l'ensemble des voisins. Dans l'étape 2 de terminaison, nous fixons la condition d'arrêt et nous commençons la boucle. La troisième étape sélectionne le premier voisin. À l'étape 4, nous allons parcourir tout l'ensemble des voisins. Dans un premier temps nous allons explorer le voisinage, en le parcourant segment par segment et en appliquant à chaque fois la méthode du Golden Section algorithme jusqu'à trouver le meilleur voisin. Dans un deuxième temps, on compare le meilleur voisin avec la solution initiale, et on retient la meilleure solution entre les deux. L'algorithme va s'arrêter une fois la condition d'arrêt est vérifiée.

7 Application et discussion des résultats

Nous avons programmé notre algorithme d'apprentissage en C++, par la suite nous l'avons appliqué sur une base de données d'Iris Data. La classification des données est connue au préalable. Comme nous allons appliquer un algorithme d'apprentissage, les données doivent être préparées afin de ressortir l'ensemble d'apprentissage, l'ensemble de test et l'ensemble d'évaluation. Ces trois ensembles devront être distincts. L'ensemble d'apprentissage est utilisé pour calculer les différents paramètres de PROAFTN. Une fois les paramètres calculés, il faut vérifier comment ils se comportent sur l'ensemble de test. Enfin, les paramètres seront testés sur l'ensemble d'évaluation.

Nous allons prendre 80% de la base de données pour l'ensemble d'apprentissage 10% pour l'ensemble de test et les 10% restante pour l'ensemble d'évaluation. Pour discuter les résultats obtenus après exécution du programme, nous allons raisonner selon les ensembles d'apprentissage de test et d'évaluation. En utilisant l'ensemble d'apprentissage et les valeurs initiales des seuils tel que proposé dans l'algorithme, soient pour S1 le minimum des évaluations par critère et par classe, et pour S2 le maximum des évaluations par critère et par classe de l'ensemble d'apprentissage. En fixant 500 comme nombre maximal des itérations nous avons obtenu nos premiers paramètres optimaux. En utilisant ces paramètres pour classer l'ensemble de test nous n'avons pas obtenu des résultats satisfaisants. Nous avons modifié la condition d'arrêt en augmentant le nombre maximal des itérations. Ceci nous a permis d'avoir les résultats avec l'ensemble de test (Tableau 1).

Étape 1. (Initialisation)

- Choisir N_k ($k=1, \dots, k_{\max}$) l'ensemble des voisins et le paramètre $k_{\max}$
- Trouver une solution initiale X_0 ($S^1_0, S^2_0, q^+_0, q^-_0$)
- Choisir un critère d'arrêt. (maxits)
- Fixer la constante α (α est le nombre d'or $= (\sqrt{5}-1)/2$)
- Choisir t la largeur finale de l'intervalle d'incertitude (la tolérance)
- Choisir maxits' notre condition d'arrêt (Nombre maximal d'itérations)

Étape 2. (Terminaison)

- Si la condition d'arrêt est vérifiée stop. (for $k=1$ until maxits do)

Étape 3. (Premier voisin)

- $k \leftarrow 1$

Étape 4. (Boucle)

- Répéter les étapes 4.1 et 4.2 jusqu'à $k=k_{\max}$

Étape 4.1. Exploration du voisinage :

- Générer aléatoirement un voisin X'' de X_0 ($X'' \in N_k(X_0)$)
- $X' \leftarrow X''$
- Appliquer GSA sur $[X', X^k]$
- $a_1 \leftarrow X''$ et $b_1 \leftarrow X^k$

Pour $j := 1$ jusqu'à maxits' faire**Étape 4.1.1:**

- Si $b_j - a_j < t$ alors $X_g \leftarrow b_j$; stop.
- Sinon, considérer x_1 et x_2 définis ci dessous et aller à l'étape 4.1.2
- $x_1 = a_j + \alpha(b_j - a_j)$
- $x_2 = b_j - \alpha(b_j - a_j)$

Étape 4.1.2:

- Si $f(x_1) < f(x_2)$ Alors $a_{j+1} \leftarrow a_j$ et $b_{j+1} \leftarrow x_2$
- Sinon $a_{j+1} \leftarrow x_1$ et $b_{j+1} \leftarrow b_j$
- $j \leftarrow j+1$ et aller à l'étape 4.1.1.

Fin pour.

- Si $f(X_g) < f(X')$, $X' \leftarrow X_g$

Étape 4.2. Bouger ou non :

- Si la solution X' est meilleure que X_0 alors $X_0 \leftarrow X'$ et aller à l'étape 3
- Sinon $k \leftarrow k+1$ et aller à l'étape 4.

Fin de la boucle 4.**Figure 7 : Algorithme d'apprentissage de détermination des paramètres de PROAFTN**

	Nombre d'itérations	Actions bien classées	Actions mal classées	Actions non classées
Ensemble de test	500	33%	7%	60%
	1000	53%	40.4%	6.6%
	2000	53%	40.4%	6.6%
	5000	60%	33.4%	6.6%

Tableau 1 : Résultats de l'ensemble de test

Nous avons donc gardé les derniers paramètres obtenus avec 5000 itérations pour classer l'ensemble d'évaluation. Nous avons obtenue alors les résultats représentés dans le tableau 2.

	Actions bien classées	Actions mal classées	Actions non classées
Ensemble d'évaluation	80%	13.4%	6.6%

Tableau 2 : Résultats de l'ensemble d'évaluation

La lecture du tableau 2 nous permet de constater que 80% des actions ont été bien classées et que uniquement 6.6% des actions ont été non classées. Ces résultats démontrent le bien fondé de notre algorithme pour automatiser la détermination des paramètres de PROAFTN ou pour d'autres applications.

8 Conclusion

L'algorithme d'apprentissage pour inférer des paramètres de PROAFTN offre plusieurs avantages. L'algorithme combine l'utilisation de deux méthodes performantes d'optimisation. La première, la technique VNS qui permet de parcourir tout l'espace des solutions réalisables, et qui saute aléatoirement d'une solution à l'autre pour éviter de rester dans un optimum local. La deuxième, le Golden Section Algorithme qui permet de déterminer l'optimum local, et qui s'adapte très bien pour notre cas d'application. La combinaison de ces méthodes nous assure le parcours complet de tout l'espace des solutions réalisables.

Il est plus facile, à notre avis de trouver des données historiques ou une ancienne base de données dans laquelle les actions sont déjà classées, que de déterminer les prototypes et les seuils en concertation avec le décideur. Grâce à la démarche proposée, l'intervention du décideur va être au niveau de la validation des paramètres obtenus par l'algorithme. Si ces paramètres ne correspondent pas à ses préférences, alors on pourra les ajuster automatiquement, jusqu'à obtenir le modèle final de classification.

Notre contribution peut être généralisée pour résoudre d'autres problèmes d'optimisation, offrant ainsi de nouvelles pistes de recherche. Nous pensons qu'il serait intéressant de valoriser notre travail en :

- généralisant l'automatisation pour les autres paramètres tels que les poids, et les seuils de discordance,
- raisonnant sur plus d'un seul prototype par classe,
- appliquant l'heuristique pour d'autres problèmes d'optimisation dans d'autres domaines d'application,
- développant une nouvelle méthode de classification basée sur l'approche multicritère et les techniques d'apprentissage.

Références

Aarts E., Lenstra J.K., *Local Search In Combinatorial Optimization*, Wiley, Chichester, 1997.

Belacel N., Boulassel M.R., "Multicriteria fuzzy assignment method: a useful tool to assist medical diagnosis", *Artif.Intell. Med*, 21 (2001), 201–207.

Belacel N., Boulassel M.R., "Multicriteria fuzzy classification procedure PROCFTN: methodology and medical application", *Fuzzy Sets and Systems*, 141- 2 (2004), 203-217.

Belacel N., *Méthodes de classification multicritère: Méthodologie et application à l'aide au diagnostic médical*, Ph.D. Thesis, Free University of Brussels, 1999.

Belacel N, Multicriteria Assignment Method PROAFTN: Methodology and Medical Application, *European Journal of Operational Research*, 125-1 (2000), 175-183.

Belacel N., Ben Abdel Aziz F., Cvetkovic A., Martel J.M., Renaud J., Urli B., "Développement de méthodes d'aide à la structuration de situations décisionnelles opérationnelles", rapport final pour le projet DND/CRDV-Visual decision, (2001).

Chase T., "Supplementary Notes On The Golden Section Search Technique", *Computer Aided Engineering*, ME 5241 (2001).

Hansen P., Mladenovic N., "Variables Neighborhood Search: Principe's And Applications", *European Journal Of operational research*, 130 (2001), 449-467.

Henriet L., *Systèmes D'évaluation Et De Classification Multicritères Pour L'aide À La Décision, Construction De Modèles Et Procédures D'affectation*, Thèse de doctorat en sciences, Université Paris Dauphine, 2000.

Moscarola J., Roy B., "Procédure Automatique D'examen De Dossiers Fondée Une Segmentation Trichotomique En Présence De Critères Multiples" *R.A.I.R.O Recherche opérationnelle*, 11 (1971), 2 :145-173.

Mousseau V., Slowinski R., "Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results", *European Journal Of operational research*, 130(2001), 263-275.

Roy B., Bouyssou D., *Aide Multicritère à la Décision*, Economica, Paris, 1993.