Codage directionnel avec attribut de longueur et optimisation de matrice de coût — Application en écriture cursive*

Marc PARIZEAU†, Nadia GHAZZALI‡ and Jean-François HÉBERT§

imprimé le 4 août 1997

Résumé

Dans cet article, nous introduisons dans un premier temps un codage directionnel avec attribut de longueur. Nous le définissons dans le cadre d'une structure de chaînes avec attribut et nous décrivons un algorithme de comparaison de chaînes basé sur ce codage. Dans un deuxième temps, à l'aide d'un algorithme génétique, nous développons une méthode générale d'optimisation pour déterminer la meilleure matrice de coût de n'importe quel algorithme de comparaison de chaînes basé sur des opérations élémentaires d'insertion, de substitution et de destruction (distance de Levenshtein). Cette méthode est ensuite testée sur des lettres cursives représentées par le codage proposé.

Mots clés: Structures de chaînes, Codage directionnel, Algorithme génétique, Reconnaissance d'écriture cursive.

1 Introduction

La représentation d'une forme en structures de chaînes est une approche bien connue en reconnaissance des formes structurelles [1]. Dans ce contexte, la comparaison d'une forme inconnue avec une forme de référence passe par la comparaison de leurs chaînes respectives. Cette comparaison peut se faire de différentes façons dont celle basée sur la distance dite de Levenshtein [2] qui consiste à compter le nombre minimum d'opérations élémentaires à effectuer sur une des chaînes pour la transformer en l'autre chaîne; les trois opérations élémentaires étant habituellement l'insertion d'un symbole, la substitution d'un symbole par un autre symbole et la destruction d'un symbole [3].

Les algorithmes de comparaison de chaînes basés sur la distance de Levenshtein nécessitent la définition d'une matrice spécifiant le coût de chacune des opérations élémentaires pour chaque combinaison de symboles. Par exemple, l'algorithme de Wagner et Fisher [4] ou encore l'algorithme de Vidal et al. [5] permettent de comparer deux chaînes, alors que l'algorithme de Myers et Miller [6] permet de comparer une chaîne avec une expression régulière. Dans cet article, nous nous intéressons au problème général de l'optimisation d'une telle matrice de coût.

La suite de cet article est subdivisée comme suit. La section 2 définit formellement le codage d'un tracé cursif en une chaîne de codes directionnels pondérés et spécifie l'algorithme utilisé pour le calcul de la distance entre de telles chaînes. La section 3 décrit la procédure d'optimisation de la matrice de coût, qui est basée sur un algorithme génétique. Finalement, les résultats expérimentaux obtenus à l'aide d'une banque de lettres cursives détachées sont présentés à la section 4.

2 Modélisation du tracé

Le but de cette section est de modéliser un tracé cursif par une structure de chaînes dont les symboles possèdent un attribut de longueur. De

*Ce travail a été présenté au 4ème Colloque National sur l'Écrit et le Document (CNED'96). Il a été supporté en partie par la subvention CRSNG de M. Parizeau et en partie par la subvention CRSNG de N. Ghazzali.

†Marc Parizeau est au département de génie électrique et de génie informatique, Université Laval, St-Foy (Québec), Canada, G1K 7P4. E-mail: *parizeau@gel.ulaval.ca*

‡Nadia Ghazzali est au département de mathématiques et de statistique, Université Laval, St-Foy (Québec), Canada, G1K 7P4. E-mail: *ghazzali@mat.ulaval.ca*

§Jean-François Hébert était au département de mathématiques et de statistique. Il est maintenant au département de génie électrique et de génie informatique, Université Laval, St-Foy (Québec), Canada, G1K 7P4. E-mail: *jfhebert@gel.ulaval.ca*

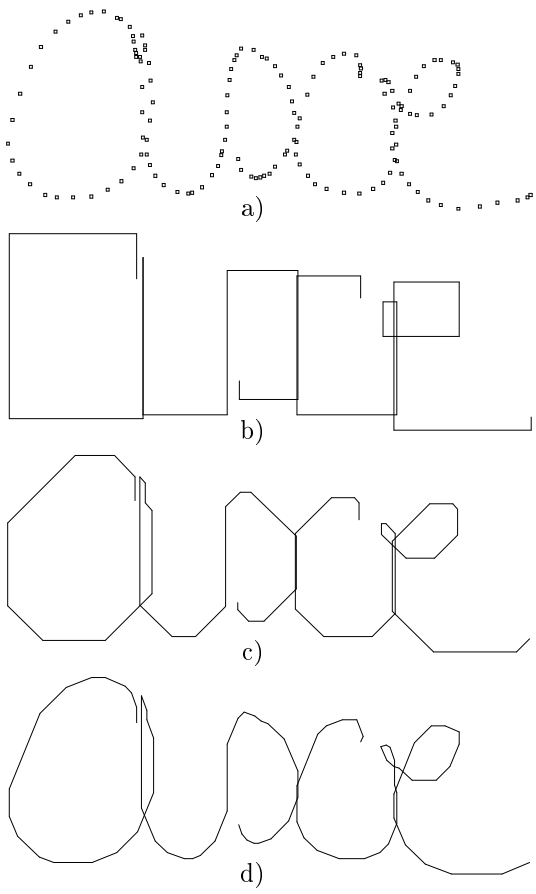

Figure 1: *Approximation polygonale d'un mot cursif : a) tracé cursif, b) tracé codé avec 4 directions, c) avec 8 directions, d) avec 16 directions.*

sorte que le problème de discrimination entre les formes revient à celui de la maximisation d'une certaine expression fonction de la distance entre deux chaînes avec attribut.

2.1 Codage directionnel avec attribut de longueur

Au départ, le tracé cursif est formé d'un ensemble de composantes[1] cursives dont chaque composante $T = p_1 p_2 \cdots p_n$ est constituée d'une suite de points $p_i = (x_i, y_i)$ échantillonnés à fréquence fixe selon les axes X et Y d'une tablette à digitaliser (voir figure 1a)). La conversion d'un tracé cursif en une chaîne de codes de direction avec attribut de longueur s'effectue en calculant d'abord la séquence des angles de la tangente au tracé. Cette tangente est approximée par l'angle des segments de droite reliant chaque point du tracé au point suivant. Ensuite, en employant un cycle d'hystérésis, les angles de cette séquence sont quantifiés sur un nombre η de niveaux et les codes consécutifs identiques sont fusionnés en un seul code dont la longueur est calculée de telle sorte que le tracé codé résultant englobe le tracé cursif tant que le sens de rotation de ce dernier reste inchangé. Les figures 1b), 1c) et 1d) illustrent le tracé codé résultant de ce processus pour le cas du tracé cursif de la figure 1a) lorsque $\eta = 4$, $\eta = 8$ et $\eta = 16$ respectivement. On remarque sur ces figures que, peu importe la valeur de η, la taille (largeur et hauteur) des boucles ainsi que la position des points d'inflexion demeurent approximativement les mêmes. Par contre, on constate que pour $\eta = 4$, le tracé codé est une version sévèrement filtrée du tracé cursif même si l'on parvient tout de même à reconnaître le mot. Il existe cependant un certain nombre de cas pathologiques pour lesquels cette reconnaissance n'est pas aussi aisée. De son côté, le tracé codé avec $\eta = 16$ reproduit à toute fin pratique parfaitement le tracé cursif (y compris les tremblements du scripteur!). Entre ces deux extrêmes le résultat obtenu avec $\eta = 8$ semble optimal car il préserve très bien la forme du tracé tout en filtrant une quantité importante d'information non pertinente à la reconnaissance.

Plus formellement, nous appelons le résultat de ce processus une *structure de chaîne avec attributs* :

Définition 1 : Une chaîne avec attributs définie sur un alphabet Σ et sur un espace vectoriel E^r de dimension r est une suite ordonnée $s_1, \ldots, s_i, \ldots, s_n$ de symboles avec attributs $s_i = (c_i; x_{i1}, \ldots, x_{ij}, \ldots, x_{ir})$ où $c_i \in \Sigma$ représente un élément structurel et x_{ij} est le $j^{\text{ème}}$ attribut associé à c_i. □

Dans le cas qui nous intéresse, chaque symbole s_i est constitué d'un couple $(c_i; l_i)$ où $c_i \in \{1, 2, \ldots, \eta\}$ représente le code de direction et $l_i \in \mathbb{N}^+$ représente la longueur de ce code. Par exemple, pour le cas du tracé codé de la figure 2 et en fixant $\eta = 8$, nous obtenons une chaîne de 12 symboles :

$$(7;19)(8;24)(1;69)(2;36)(3;24)(4;35)(5;39)(6;63)$$
$$(7;67)(8;52)(1;73)(2;35)$$

dont l'unité de longueur des l_i correspond aux unités de la tablette à digititaliser c'est-à-dire le millième de pouce.

L'avantage des structures de chaînes avec attribut est, d'une part, qu'elles sont plus com-

[1] Une composante cursive est définie par la portion du tracé cursif compris entre la pose du crayon sur la surface d'écriture et la prochaine levée de crayon [7].

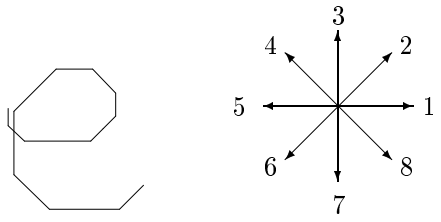

Figure 2: *Tracé codé en a) à l'aide des huit directions représenté en b).*

pactes que les structures de chaînes conventionnelles et, d'autre part, qu'elles ne requièrent pas de compromis au niveau du choix de la longueur unitaire. En effet, pour l'exemple de la figure 2, une chaîne conventionnelle utilisant la même unité de longueur requerrait 536 codes de direction soit la somme des longueurs des 12 symboles avec attribut. Évidemment, le choix d'une plus grande unité de longueur atténuerait ce problème mais engendrerait par ailleurs des erreurs d'arrondi lorsque, par exemple, la taille des boucles dans l'écriture approcherait cette longueur unitaire.

Pour comparer deux tracés cursifs nous procédons donc à la comparaison de leur tracé codé en structure de chaînes avec attribut. Cette comparaison est basée sur la distance dite de Levenshtein [2] consistant à calculer le nombre minimum d'opérations d'insertion, de substitution et de destruction requises pour transformer la première chaîne en la deuxième. L'algorithme de comparaison que nous proposons ici est une adaptation de l'algorithme bien connu de Wagner et Fisher [4] qui nécessite la définition d'une matrice de coût associée aux différentes opérations élémentaires.

2.2 Propriétés de la matrice de coût

Soit $\gamma_\Sigma : (\Sigma \cup \{\lambda\})^2 \longrightarrow I\!R^+$ la fonction associée à cette matrice de coût avec Σ représentant l'alphabet de la structure de chaînes et λ le symbole blanc. Alors, $\gamma_\Sigma(\lambda, c)$ représente le coût d'insertion d'un code c, $\gamma_\Sigma(c, \lambda)$ le coût de destruction d'un code c et $\gamma_\Sigma(c_1, c_2)$ le coût de substitution d'un code c_1 par un code c_2. Pour remplir toutes les conditions d'une distance, γ_Σ doit posséder les propriétés suivantes :

1. $\forall (i,j) \in (\Sigma \cup \{\lambda\})^2,\ \gamma_\Sigma(i,j) \geq 0$: les coûts d'insertion, de destruction et de substitution doivent être positifs ou nuls pour que la distance entre deux chaînes soit toujours positive ou nulle ;
2. $\forall (i,j) \in (\Sigma \cup \{\lambda\})^2,\ \gamma_\Sigma(i,j) = 0 \Longleftrightarrow i = j$: γ_Σ est à diagonale nulle et, par conséquent, le coût de substitution d'un caractère par lui-même est nul. On obtient ainsi une distance nulle entre deux chaînes identiques.
3. $\forall (i,j) \in (\Sigma \cup \{\lambda\})^2,\ \gamma_\Sigma(i,j) = \gamma_\Sigma(j,i)$: γ_Σ est symétrique pour que la distance entre deux chaînes x et y soit la même qu'entre y et x ;
4. $\forall (i,j,k) \in (\Sigma \cup \{\lambda\})^3,\ \gamma_\Sigma(i,k) \leq \gamma_\Sigma(i,j) + \gamma_\Sigma(j,k)$: l'inégalité triangulaire permet d'assurer que certaines substitutions ne sont pas éliminées en raison de leur coûts trop élevés. Par exemple, si le coût pour substituer $i \rightarrow j$ est plus élevé que la somme des coûts pour insérer $\lambda \rightarrow j$ et pour détruire $i \rightarrow \lambda$, alors la substitution directe ne sera jamais réalisée car il sera toujours plus avantageux d'effectuer une destruction suivie d'une insertion.

Notons que ces quatre propriétés font passer le nombre de paramètres distincts de la matrice de coût de $|\gamma_\Sigma| = (|\Sigma|+1)^2$ à $[(|\Sigma|+1) \times |\Sigma|]/2$.

2.3 Algorithme de comparaison

Soit deux chaînes avec attribut χ et ψ :

$$\chi = s_1 s_2 \cdots s_i \cdots s_m = \chi(m) \qquad (1)$$
$$\psi = s_1 s_2 \cdots s_j \cdots s_n = \psi(n) \qquad (2)$$

avec $s_i = (c_i, l_i)$ et $s_j = (c_j, l_j)$ représentant les $i^{\text{ème}}$ et $j^{\text{ème}}$ symboles avec attribut de χ et ψ respectivement; et $\chi(i)$ et $\psi(j)$ dénotant les sous-chaînes $s_1 \cdots s_i$ de χ et $s_1 \cdots s_j$ de ψ. Alors, la distance $D(\chi, \psi)$ entre les chaînes χ et ψ est obtenue par $D[\chi(m), \psi(n)] = D(m,n)$ et l'expression de $D(i,j)$ est donnée à l'équation (3) avec $\overline{l}_\chi = \frac{1}{m}\sum_{i=1}^{m} l_i$ et $\overline{l}_\psi = \frac{1}{n}\sum_{j=1}^{n} l_j$ correspondant respectivement aux longueurs moyennes des codes des chaînes χ et ψ.

L'équation 3 exprime la distance $D(i,j)$ entre les sous-chaînes $\chi(i)$ et $\psi(j)$ à l'aide d'une formule de récurrence dont le résultat est le minimum de trois expressions qui dépendent respectivement de $D(i, j-1)$, $D(i-1, j-1)$ et de $D(i-1, j)$. La première expression correspond à la distance $D(i, j-1)$ additionnée du coût de l'insertion de s_j dans χ. Le terme $\gamma_\Sigma(\lambda, c_j)$ doit être interprété comme le coût d'insertion d'un code c_j de longueur moyenne et le ratio $l_j/\overline{l}_\psi$ correspond à la longueur effective de c_j normalisée par la longueur moyenne

$$D(i,j) = \min \begin{cases} D(i,j-1) + \frac{l_j}{\overline{l}_\psi}\gamma_\Sigma(\lambda, c_j) \\ D(i-1,j-1) + \begin{cases} \left(1 + \left|\frac{l_i}{\overline{l}_\chi} - \frac{l_j}{\overline{l}_\psi}\right|\right)\gamma_\Sigma(c_i, c_j) & \text{Si } c_i \neq c_j \\ \left(\frac{l_i}{\overline{l}_\chi} - \frac{l_j}{\overline{l}_\psi}\right)\gamma_\Sigma(c_i, \lambda) & \text{Si } c_i = c_j \text{ et } \frac{l_i}{\overline{l}_\chi} \geq \frac{l_j}{\overline{l}_\psi} \\ \left(\frac{l_j}{\overline{l}_\psi} - \frac{l_i}{\overline{l}_\chi}\right)\gamma_\Sigma(\lambda, c_j) & \text{Si } c_i = c_j \text{ et } \frac{l_i}{\overline{l}_\chi} < \frac{l_j}{\overline{l}_\psi} \end{cases} \\ D(i-1,j) + \frac{l_i}{\overline{l}_\chi}\gamma_\Sigma(c_i, \lambda) \end{cases} \quad (3)$$

des codes dans ψ. La deuxième expression correspond à la distance $D(i-1, j-1)$ additionnée du coût de la substitution de s_i de χ par s_j de ψ. L'expression de ce coût de substitution dépend de trois cas distincts. Le premier cas est lorsque $c_i \neq c_j$ c'est-à-dire lorsque l'on doit substituer deux codes différents. Alors, le terme $\gamma_\Sigma(c_i, c_j)$ doit être interprété comme le coût de substitution de deux codes c_i et c_j de longueur moyenne et le facteur $(1 + |l_i/\overline{l}_\chi - l_j/\overline{l}_\psi|)$ mesure la différence de leur longueur respective. Le deuxième cas est lorsque $c_i = c_j$ et que la longueur normalisée de c_i est plus grande ou égale à celle de c_j. Alors il s'agit simplement de calculer le coût de destruction d'un c_i de longueur normalisée $(l_i/\overline{l}_\chi - l_j/\overline{l}_\psi)$. Le troisième cas est le symétrique du deuxième c'est-à-dire que si $c_i = c_j$ et que $(l_i/\overline{l}_\chi < l_j/\overline{l}_\psi)$, alors il s'agit de calculer le coût d'insertion d'un c_j dont la longueur normalisée égale la différence des longueurs normalisées de c_i et c_j. Si γ_Σ est symétrique, ces deux derniers cas peuvent être fusionnés en un seul en prennant la valeur absolue. Finalement, la troisième expression de l'équation 3 s'interprète de manière similaire à la première mais en détruisant cette fois s_i dans χ.

3 Optimisation de la matrice de coût

Le choix des valeurs de la matrice de coût constitue un problème majeur pouvant affecter grandement le résultat de la comparaison de structures de chaînes. En pratique, cette matrice de coût est souvent fixée plus ou moins arbitrairement en tenant compte, lorsque ceci est possible, des statistiques de confusion entre les symboles. Cependant, ces statistiques sont habituellement difficiles à évaluer et, d'ailleurs, ne garantissent aucunement une performance optimale. Dans cette section, nous proposons une alternative simple et efficace permettant de déterminer une matrice de coût s'approchant de la matrice optimale grâce à l'emploi d'un algorithme génétique.

3.1 Algorithmes génétiques

Parmi les méthodes d'optimisation guidées aléatoirement, les algorithmes génétiques [8, 9] sont bien adaptés aux problèmes dont la fonction d'évaluation ne possède pas de forme analytique. En effet, dans le cas qui nous intéresse, nous cherchons à optimiser le résultat d'un algorithme dont la forme récursive est particulièrement mal adaptée aux méthodes d'optimisation fondées sur le calcul.

Les algorithmes génétiques sont basés sur deux grands principes empruntés, d'une part, à la théorie de l'évolution par la sélection naturelle (Darwin, 1859) et, d'autre part, au domaine de la reproduction cellulaire. Le principe de la sélection naturelle prévoit que les individus les plus forts ont plus de chances de survie et qu'ils ont, par conséquent, plus d'opportunités de se reproduire et de transmettre leur bagage génétique de génération en génération. Le principe de la reproduction cellulaire consiste quant à lui à engendrer de nouveaux individus à partir de trois opérations fondamentales : la reproduction, la mutation et le croisement. L'opération de reproduction consiste simplement à dédoubler le bagage génétique d'un individu pour créer un clone alors que les opérations de mutation et de croisement permettent respectivement de modifier le code génétique d'un individu ou encore d'échanger une partie du code génétique de deux individus pour ainsi engendrer de la diversité dans une population.

Bien qu'il existe de nombreuses variantes dans les algorithmes génétiques, elles partagent essentiellement les trois grandes étapes suivantes :

1. Génération d'une population initiale de taille N où les individus sont constitués d'un seul chromosome formé par la concaténation de l'ensemble des paramètres du système, cha-

cun étant quantifié sur un certain nombre de bits. Pour chaque individu, évaluer sa force à l'aide de la fonction d'évaluation du système ;

2. À partir de la population précédente, engendrer par reproduction une nouvelle population de même taille en privilégiant les individus les plus forts. À l'intérieur de cette nouvelle population, appliquer sur certains individus les opérateurs de mutation et de croisement (avec probabilités p_m et p_c respectivement) pour créer de la diversité ;
3. Recommencer l'étape 2 tant que le nombre de générations simulées n'est pas suffisant pour obtenir un optimum.

Il nous reste maintenant à définir une fonction d'évaluation capable de mesurer la "force" de nos individus c'est-à-dire le pouvoir discriminant d'un algorithme de comparaison de chaînes avec matrice γ_Σ fixée.

3.2 Fonction d'évaluation

Soit $\mathcal{C}$ un ensemble de p classes d'objets :

$$\mathcal{C} = \{C_1, \ldots, C_i, \ldots, C_p\} \tag{4}$$

où chaque classe C_i est représentée par un ensemble de q_i structures de chaînes (avec ou sans attribut) :

$$C_i = \{\chi_{i1}, \ldots, \chi_{ij}, \ldots, \chi_{iq_i}\} \tag{5}$$

où χ_{ij} est le $j^{\text{ème}}$ prototype de la classe C_i.

Pour trouver la matrice de coût γ_Σ d'un algorithme de comparaison de chaînes, nous proposons d'employer la fonction d'évaluation F suivante qui estime le taux de reconnaissance d'un classifieur gaussien qui minimise l'erreur :

$$F = \sum_{i=1}^{p} \sum_{\substack{k=1 \\ k \neq i}}^{p} (1 - \hat{\epsilon}_{ik}) \tag{6}$$

où $\hat{\epsilon}_{ik}$ représente l'erreur estimée du classifieur lorsque l'on veut discriminer entre la classe i et la classe k (voir figure 3) :

$$\hat{\epsilon}_{ik} = \Phi\left[\frac{-(\bar{D}_{ik} - \bar{D}_{ii})}{2S_{ik}}\right] \tag{7}$$

où Φ est la fonction de répartition d'une loi normale $N(0,1)$, $\bar{D}_{ii}$ et $\bar{D}_{ik}$ représentent respectivement les moyennes des distances intra et inter classes, et S^2_{ik} correspond à la variance groupée des distances intra et inter classes. Notez que l'argument de la fonction de répartition Φ à l'équation

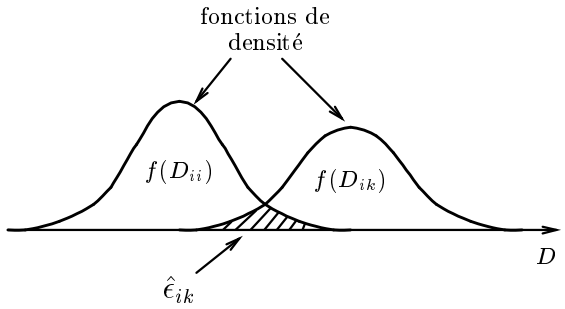

Figure 3: *Erreur estimée de classement pour les classes i et k.*

(7) correspond à la distance de Mahanalobis dans le cas unidimensionnel [10]. Notez également que si $\bar{D}_{ik} \geq \bar{D}_{ii}$ alors $0 \leq \hat{\epsilon}_{ik} \leq 0.5$. L'expression des moyennes $\bar{D}_{ii}$ et $\bar{D}_{ik}$ est donnée par :

$$\bar{D}_{ii} = \sum_{j=1}^{q_i} \sum_{\substack{l=1 \\ l \neq j}}^{q_i} \frac{D(\chi_{ij}, \chi_{il})}{q_i(q_i - 1)} \tag{8}$$

$$\bar{D}_{ik} = \sum_{j=1}^{q_i} \sum_{l=1}^{q_k} \frac{D(\chi_{ij}, \chi_{kl})}{q_i q_k} \tag{9}$$

et la variance groupée des distances s'obtient par :

$$S_{ik} = \sqrt{\frac{[q_i(q_i - 1) - 1]s^2_{ii} + [q_i q_k - 1]s^2_{ik}}{[q_i(q_i - 1) + q_i q_k - 2]}} \tag{10}$$

où s^2_{ii} et s^2_{ik} sont respectivement les variances intra et inter classes exprimées par :

$$s^2_{ii} = \sum_{j=1}^{q_i} \sum_{\substack{l=1 \\ l \neq j}}^{q_i} \frac{[D(\chi_{ij}, \chi_{il}) - \bar{D}_{ii}]^2}{q_i(q_i - 1) - 1} \tag{11}$$

$$s^2_{ik} = \sum_{j=1}^{q_i} \sum_{l=1}^{q_k} \frac{[D(\chi_{ij}, \chi_{kl}) - \bar{D}_{ik}]^2}{q_i q_k - 1} \tag{12}$$

4 Résultats expérimentaux

Pour tester notre méthode d'optimisation, nous avons utilisé une banque de lettres cursives contenant des échantillons des 26 lettres ($|\mathcal{C}| = p = 26$ classes) minuscules de l'alphabet romain produits par 10 scripteurs différents. Chaque scripteur ayant tracé dix fois chacune des lettres en employant son écriture naturelle, la banque contient 100 prototypes par classe pour un total de 2600 prototypes.

De ces 100 prototypes par classe, nous en avons choisi aléatoirement 10 pour procéder à une

	λ	1	2	3	4	5	6	7	8
λ	0	β	β	β	β	β	β	β	β
1	β	0	α_1	α_2	α_3	α_4	α_3	α_2	α_1
2	β	α_1	0	α_1	α_2	α_3	α_4	α_3	α_2
3	β	α_2	α_1	0	α_1	α_2	α_3	α_4	α_3
4	β	α_3	α_2	α_1	0	α_1	α_2	α_3	α_4
5	β	α_4	α_3	α_2	α_1	0	α_1	α_2	α_3
6	β	α_3	α_4	α_3	α_2	α_1	0	α_1	α_2
7	β	α_2	α_3	α_4	α_3	α_2	α_1	0	α_1
8	β	α_1	α_2	α_3	α_4	α_3	α_2	α_1	0

Figure 4: *Matrice de coût à optimiser.*

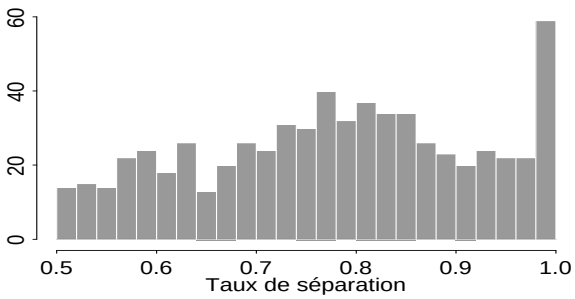

Figure 5: *Histogramme de la distribution des termes* $(1 - \epsilon_{ik})$ *(*$26 \times 25 = 650$ *arrangements).*

expérience d'optimisation ($\forall i, |C_i| = q_i = 10$) impliquant pour chaque évaluation de la fonction F, le calcul de $p(q_i^2 - q_i) = 2\,340$ distances intra-classes et $(p^2 - p)q_i^2 = 65\,000$ distances inter-classes, pour un grand total de 67 340 comparaisons de chaînes.

Pour le choix du codage, nous avons fixé $\eta = 8$ c'est-à-dire un codage à huit directions. En respectant les propriétés d'une distance, il découle de ce choix une matrice de coût pouvant contenir au plus $9 \times 8/2 = 36$ paramètres distincts (§2.2). Pour diminuer la dimension de l'espace de recherche, nous avons réduit ce nombre à seulement 5 paramètres en posant les restrictions supplémentaires suivantes : 1) les coûts d'insertion et de destruction pour tous les symboles de l'alphabet sont identiques et valent β, 2) les coûts de substitution ne dépendent que l'angle θ_{ij} entre les deux directions i et j que l'on veut substituer :

$$\begin{array}{llll} \text{Si} & \theta_{i,j} = \pi/4 & \text{alors} & \gamma_\Sigma(i,j) = \alpha_1 \\ \text{Si} & \theta_{i,j} = \pi/2 & \text{alors} & \gamma_\Sigma(i,j) = \alpha_2 \\ \text{Si} & \theta_{i,j} = 3\pi/4 & \text{alors} & \gamma_\Sigma(i,j) = \alpha_3 \\ \text{Si} & \theta_{i,j} = \pi & \text{alors} & \gamma_\Sigma(i,j) = \alpha_4 \end{array}$$

Ces restrictions ne sont pas nécessaires au bon fonctionnement de la méthode proposée, mais permettent simplement de réduire la dimension de l'espace des paramètres qui, pour n'importe quel système, influence grandement la complexité de son optimisation. Ainsi paramétrisée, notre matrice de coût est illustrée à la figure 4.

Notre algorithme génétique engendre des populations de taille $N = 25$ en quantifiant les paramètres de la matrice sur 10 bits et en utilisant un opérateur de mutation avec probabilité $p_m = 0.02$ ainsi qu'un opérateur de croisement avec probabilité $p_c = 0.35$. Le domaine de définition des paramètres est limité à la plage $[0, 5]$. Après 500 générations, l'individu le plus fort de la population (obtenu à la 85 ième génération) possède les paramètres $\beta = 0.70$, $\alpha_1 = 1.02$, $\alpha_2 = 2.37$, $\alpha_3 = 4.79$ et $\alpha_4 = 3.50$ qui lui confère une force $F = 506.5$. Sachant que F est bornée à $26 \times 25 = 650$, on peut aisément interpréter $F/650 = 77.9\%$ comme un taux moyen de séparation entre toutes les paires de classes. En comparaison, une matrice de coût unitaire (i.e. $\beta = \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$) obtient un taux de 75.8% soit une différence absolue de 2.1%.

La figure 5 donne pour l'individu le plus fort, l'histogramme de la distribution de ses taux de séparation entre toutes les paires de classes. Cette figure montre que certaines paires de classes posent problèmes. Il importe de bien comprendre que ceci dépend des données et ne constitue nullement une faiblesse ni de l'algorithme de comparaison, ni de la méthode d'optimisation. En effet, les lettres cursives associées à nos classes contiennent assez souvent des styles d'écriture complètement différents qui engendrent des distances intra-classes équivalentes (ou même supérieures !) aux distances inter-classes. C'est le cas, par exemple, de nos échantillons de lettres 'b' qui comportent deux formes radicalement différentes: .

La figure 6 montre l'ensemble des lettres cursives qui ont servi à cette expérience. Dans un système de reconnaissance, il s'agirait bien entendu de sélectionner adéquatement plusieurs prototypes par classe et d'utiliser un classifieur de type "plus proche voisin".

Pour vérifier que la performance de notre meilleur individu n'est pas un fruit du hasard, nous avons engendré aléatoirement 20 nouveaux échantillons, tous de taille 260 (10 lettres par classe), pour lesquels nous avons évalué la fonction d'adéquation F en employant successivement notre matrice optimale puis la matrice unité. Les résultats sont donnés sous forme de graphique à la figure 7. En moyenne, le taux de séparation

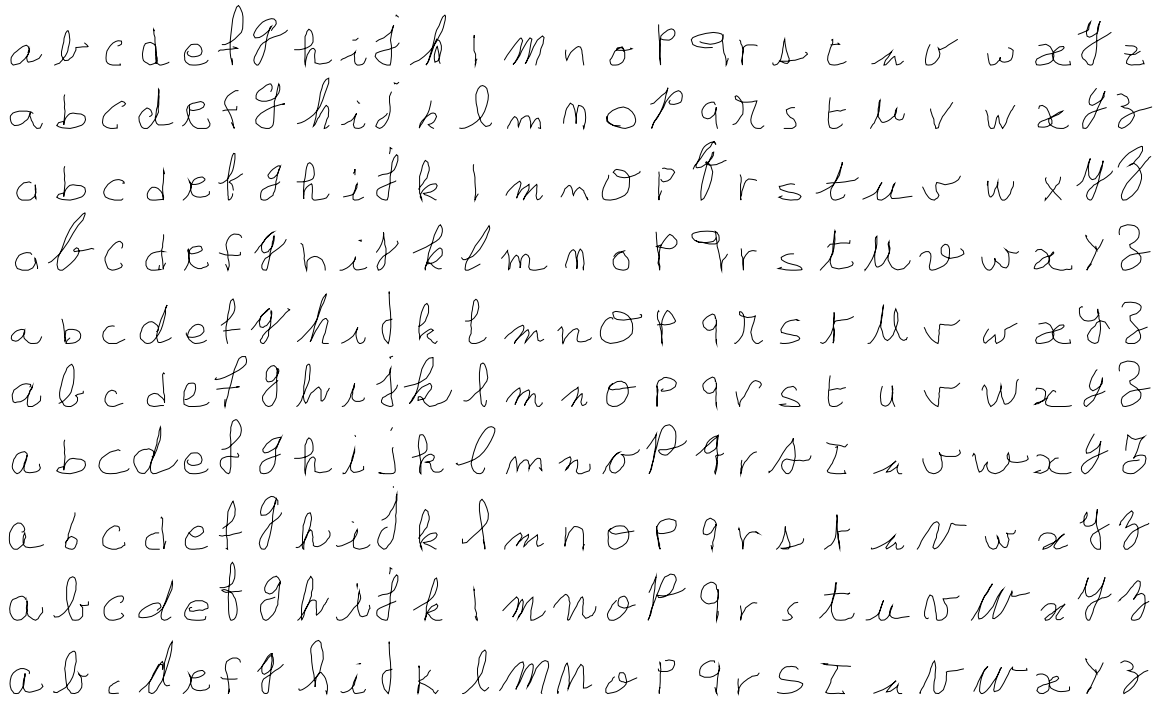

Figure 6: *Échantillon des lettres ayant servi à l'optimisation de la matrice de coût.*

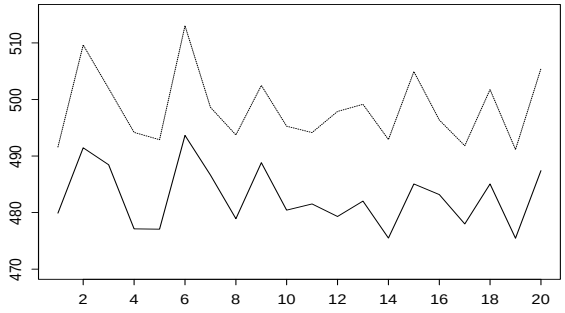

Figure 7: *Comparaison des valeurs de F de 20 échantillons aléatoires pour la matrice optimale (ligne pointillée) et la matrice unité (ligne pleine).*

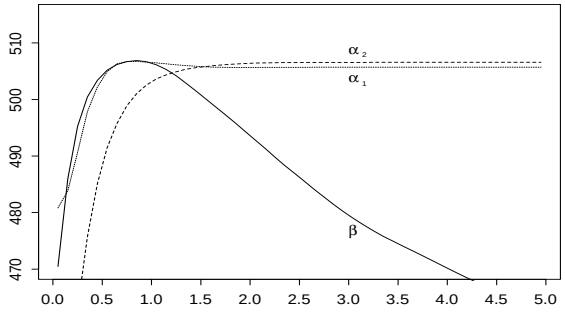

Figure 8: *Les valeurs de F en fonction respectivement des paramètres α_1, α_2 et β, les autres étant fixes.*

obtenu avec la matrice optimale est de 2.4% supérieur à celui de la matrice unité, et ceci avec un faible écart-type de 0.38%, ce qui confirme que cette différence de performance est tout à fait significative.

Finalement, le graphique de la figure 8 montre le comportement de F lorsque nous faisons varier un paramètre à la fois (α_1, α_2 et β) de notre individu optimal. On observe que la valeur maximale $F = 506.5\%$ est bien un optimum, du moins localement lorsqu'un seul paramètre varie. On observe également qu'à partir d'un certain seuil, F semble devenir indépendant de α_1 et α_2. Ce seuil correspond en fait à 2β et s'explique facilement par le fait que lorsque $\alpha_i > 2\beta$, la substitution représentée par α_i ne sera plus effectuée par l'algorithme car il deviendra plus économique d'effectuer une destruction suivie d'une insertion que d'effectuer une substituti que pour notre matrice de coût optimale, seules les substitutions de deux symboles associés à des directions adjacentes sont permises et que le coût de ces substitutions est

égal à $\sqrt{2}$ fois le coût d'insertion/destruction (i.e. $\beta^2 + \beta^2 = \alpha_1^2$).

5 Conclusion

Nous avons introduit dans la première partie de cet article, une représentation du tracé cursif basée sur un codage directionnel avec attribut de longueur, ainsi qu'un algorithme pour mesurer la distance de Levenshtein entre deux tracés ainsi codés. Dans la seconde partie, nous avons proposé une méthode basée sur les algorithmes génétiques, pour l'optimisation de la matrice de coût employée par n'importe quel algorithme de comparaison de structures de chaînes. Plus spécifiquement, nous avons défini une fonction d'évaluation mesurant la séparation inter-classes qu'engendre l'algorithme de comparaison de chaînes ; la matrice de coût optimale étant celle qui maximise cette séparation. Cette méthode d'optimisation a été testée sur des lettres cursives représentées par le codage introduit dans la première partie.

References

[1] L. Miclet, *Méthodes structurelles pour la reconnaissance des formes*, Eyrolles, 1984.

[2] A. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", *Sov. Phy. Dohl.*, vol. 10, pp. 707-710, février 1966.

[3] D. Sankoff, J.B. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983.

[4] R.A. Wagner, M.J. Fischer, "The String-to-String Correction Problem", *Journal of the ACM*, vol. 21, no. 1, pp. 168-173, 1974.

[5] E. Vidal, A. Marzal, P. Aibar, "Fast Computation of Normalized Edit Distances", *IEEE trans. on Pattern Anal. and Machine Intel.*, vol. PAMI-17, no. 9, pp. 899-902, 1995.

[6] E.W. Myers, W. Miller, "Approximate Matching of Regular Expressions", *Bulletin of Mathematical Biology*, vol. 51, no. 1, pp. 5-37, 1989.

[7] R. Plamondon, F.J. Maarse, "An evaluation of Motor Models of Handwrting", *IEEE trans. on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1060-1072, 1989.

[8] Z. Michalewick, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

[9] D. E. Goldberg, *Algorithmes génétiques : exploration, optimisation et apprentissage automatique*, Addison-Wesley (France), 1994.

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.