Math Appl

Lundi 8 avril 2002 à 17 h

Génétique et informatique

Bernard Vuilleumier

http://Hypatie.ge.ch

Centre pédagogique des technologies de l'information et de la communication (CPTIC)
Rue Théodore-de-Bèze 2
Case Postale 3144
1211 GENÈVE 3
Tél: (022) 318.05.30
Fax: (022) 318.05.35
Directeur: Raymond Morel

Lettre nº 169

L'évolution a donné lieu à une extraordinaire diversité de formes vivantes aux capacités étonnantes. Différents organismes, adaptés aux conditions particulières de leur environnement, forment des populations coopérantes ou en compétition qui évoluent au gré de variations et de la sélection. Le plan de la croissance des organismes, « encodé » dans les génomes, varie de génération en génération. Les individus qui finissent par prévaloir sont ceux dont les capacités spécifiques s'accordent le mieux aux conditions environnementales. Voilà, très résumés, les principes de l'évolution et de l'adaptation des formes vivantes sur Terre. Que l'on soit d'accord ou non avec cette théorie, on peut s'en inspirer pour développer des concepts et des stratégies informatiques aptes à résoudre des tâches d'apprentissage et à traiter des problèmes d'optimisation pour les systèmes intelligents artificiels. Parmi ces approches, les plus fascinantes sont celles qui utilisent les algorithmes et la programmation génétique. Elles jouent un rôle croissant comme alternative à la programmation « classique » et comptent déjà à leur actif de nombreux succès, notamment en matière de robotique et d'intelligence artificielle.

Les algorithmes génétiques s'inspirent des mécanismes naturels de l'évolution

Les *algorithmes génétiques* sont utilisés pour simuler des processus évolutifs. Ils s'inspirent des mécanismes naturels de reproduction des cellules d'individus sexués. Ils adoptent quatre principes utilisés par la nature:

- séparation de l'information génétique et de son expression phénotypique;
- encodage discret de l'information génétique;
- recombinaison due à la reproduction sexuée;
- organisation modulaire des entités utilisées.

La programmation génétique recourt au hasard pour générer des programmes!

La *programmation génétique* recourt au hasard pour générer des programmes. Quiconque a programmé au moins une fois dans sa vie ne peut être que sceptique à l'égard de ce procédé: une virgule mal placée, un point virgule oublié et un programme ne « tourne» plus! Comment dès lors une approche générant du code plus ou moins aléatoirement pourrait-elle permettre d'obtenir des programmes syntaxiquement corrects et exécutables pour simuler des systèmes évolutifs? La solution passe par une représentation adéquate des programmes. Si nous parvenons à représenter la structure d'un programme en entités modulaires telles que « fonction», « sous-programme», « variable», « déclaration», «séquence d'instructions », « boucle», « condition», etc, l'idée de créer un programme en recourant à un mécanisme évolutif prend tout son sens.

Les algorithmes génétiques représentent un génome à l'aide d'une structure linéaire…

La différence majeure entre algorithme génétique et programmation génétique réside dans la différence de structure des programmes associés. Les algorithmes génétiques encodent l'information sous forme linéaire alors que la programmation génétique fait usage d'une structure en arbre pour représenter un génome. Dans un algorithme génétique, les unités qui mutent et qui se recombinent ne sont pas des caractères ou des séquences d'instructions mais des modules fonctionnels, ce qui offre des possibilités totalement nouvelles pour la modification d'un programme à l'aide d'opérateurs génétiques d'évolution. En principe, il devient possible de faire évoluer un programme en taille, en forme et en contenu sémantique aussi bien qu'algorithmique! Certains biologistes considèrent même que l'introduction d'entités structurées en arbre pour représenter un génome constitue une percée essentielle de la recherche dans le domaine de la vie artificielle.

Alors que la programmation génétique fait usage d'une structure en arbre

Prochaine réunion: lundi 6 mai 2002 à 17h.

Travaux pratiques

Mots clefs
Algorithme, génétique, informatique, programmation.

Pour introduire les notions de base d'un algorithme génétique

Exercice 1
a) Construisez une représentation $s = \{s_1, \ldots s_n\}$ d'un chromosome constitué d'une séquence de gènes s_i, chaque gène pouvant prendre une valeur A_i dans un alphabet fini.
b) Modifiez la représentation s de manière à obtenir l'indice de position des gènes en plus de leurs valeurs $s = \{\{1, s_1\}, \ldots \{n, s_n\}\}$
c) Construisez une représentation $s^{(2)} = \{\{1, \{s_{11}, s_{21}\}\}, \ldots \{n, \{s_{1n}, s_{2n}\}\}$ d'une paire de chromosomes.
d) Visualisez graphiquement ces « chromosomes » en représentant chaque gène par un carré et en utilisant différents niveaux de gris pour représenter les allèles.

Pour illustrer le mécanisme de traduction des codons en ADN

Exercice 2
Les gènes représentent les unités de base de l'information génétique stockée dans l'acide désoxyribonucléique (ADN) des cellules. Chez tous les organismes, les mêmes quatre bases nucléiques – adénine (A), cytosine (C), guanine (G) et uracil (U) – forment des sous-unités de stockage de l'information et constituent l'« alphabet » permettant de décrire les gènes de l'ADN. Un groupe de trois nucléotides définit un « mot ». Un gène consiste en une séquence de plusieurs centaines de ces « mots» encodant la structure d'une protéine. La transformation des « codons » (triplets) en une séquence d'acides aminés est accomplie par les ribosomes.
a) Construisez une représentation d'un chromosome en utilisant l'alphabet {A, C, G, U}.
b) Générez une table de toutes les combinaisons possibles de triplets sur cet alphabet. Sachant qu'on ne trouve que 20 acides aminés dans la nature, que peut-on conclure?
c) Créez un chromosome comportant quelques centaines de bases nucléiques.
d) Réunissez ces bases par groupes de trois et « décodez» les « mots» afin d'obtenir une séquence d'acides aminés en utilisant les labels donnés en marge, puis les noms ci-dessous:

```
tripletToLabel =
  {{G, C, _} → "Ala",
   {U, G, U | C} → "Cys",
   {G, A, U | C} → "Asp",
   {G, A, A | G} → "Glu",
   {U, U, U | C} → "Phe",
   {G, G, _} → "Gly",
   {C, A, U | C} → "His",
   {A, U, U | C | A} → "Ile",
   {A, A, A | G} → "Lys",
   {U, U, A | G} → "Leu",
   {C, U, _} → "Leu",
   {A, U, G} → "Met",
   {A, A, U | C} → "Asn",
   {C, C, _} → "Pro",
   {C, A, A | G} → "Gln",
   {C, G, _} → "Arg",
   {A, G, A | G} → "Arg",
   {U, C, _} → "Ser",
   {A, G, U | C} → "Ser",
   {A, C, _} → "Thr",
   {G, U, _} → "Val",
   {U, G, G} → "Trp",
   {U, A, U | C} → "Tyr",
   {U, A, A | G} → "STOP",
   {U, G, A} → "STOP"
  };
```

```
labelToAminoAcid =
  {"Ala" Æ "Alanine",
   "Arg" Æ "Arginine",
   "Asn" Æ "Asparagine",
   "Asp" Æ "Asparatic Acid",
   "Cys" Æ "Cysteine",
   "Gln" Æ "Glutamine",
   "Glu" Æ "Glutamic Acid",
   "Gly" Æ "Glycine",
   "His" Æ "Histidine",
   "Ile" Æ "Isoleucine",
   "Leu" Æ "Leucine",
   "Lys" Æ "Lysine",
   "Met" Æ "Methionine/START",
   "Phe" Æ "Phenylalanine",
   "Pro" Æ "Proline",
   "Ser" Æ "Serine",
   "Thr" Æ "Threonine",
   "Trp" Æ "Tryptophan",
   "Tyr" Æ "Tyrosine",
   "Val" Æ "Valine"
  };
```

Sources et bibliographie

• Christian Jacob, *Illustrating Evolutionary Computation with Mathematica*, Academic Press, (2001), 578 pp.
• http://www.cpsc.ucalgary.ca/njacob/IEC/