Prof. Dr. Gerhard Weiss
Department of Knowledge Engineering
Maastricht University

Einführung in die Künstliche Intelligenz

— Übungsblatt 4 —

Übung 4-1 (Evolutionäre Algorithmen)

Bereits seit vielen Jahren werden innerhalb der Künstlichen Intelligenz sogenannte *evolutionäre Algorithmen* – je nach Ausprägung auch *genetische Algorithmen* oder *Evolutionsstrategien* genannt – intensiv untersucht. Es handelt sich dabei um allgemeine, biologisch inspirierte Optimierungsverfahren, die auf den Prinzipien der Evolution – Mutation, Rekombination und Selektion – basieren. Diese Verfahren haben sich insbesondere in komplexen und unbekannten Anwendungsfeldern bewährt; erwähnt sei hier beispielsweise die Optimierung von Schedules (z.B. Maschinenbelegung), die Optimierung von Parametern und Simulationsmodellen, die Evolution von Proteinen und die Bestimmung von Gleichgewichtszuständen in dynamischen Systemen. Die prinzipielle Arbeitsweise der evolutionären Algorithmen ist sehr einfach und kann – ohne näher auf den biologischen Hintergrund einzugehen – wie folgt erklärt werden. Evolutionäre Algorithmen arbeiten stets auf Mengen (sog. Populationen) von Strukturen (sog. Genotypen oder Individuen). Jeder dieser Genotypen kodiert einen bestimmten Lösungsvorschlag (auch Phänotyp genannt) für das zugrunde liegende Optimierungsproblem, wobei jedem Lösungsvorschlag eine gewisse Güte oder Fitness zugeordnet ist (je besser der Vorschlag um so höher ist seine Fitness). Mit Hilfe von evolutionären Operatoren – Selektion, Rekombination und Mutation – wird eine neue Populationen erzeugt; dies wird so oft iteriert, bis eine Population einen Phänotypen enthält, der "eine genügend hohe Fitness" aufweist. Die Abbildung 4-1 zeigt eine allgemeine Formulierung des evolutionären Algorithmus in Pseudonotation. Im folgenden werden am Beispiel des Traveling Salesman Problems (TSP) wichtige Details beziehungsweise Parameter zur Realisierung dieses Algorithmus angegeben.

Aufgabe beim TSP: ein Verkäufer soll vorgegebene Städte derart besuchen, dass seine Wegstrecke minimal ist und jede Stadt genau einmal besucht wird. (Wir fordern hier nicht, dass die erste und letzte besuchte Stadt identisch sind.) Im Rahmen dieser Übung betrachten wir ein 30-Städte-TSP, wobei die [x,y]-Koordinaten der einzelnen Städte wie folgt sind (auf genaue Eingabe der Koordinaten achten!): [82,7], [91,38], [83,46], [71,44], [64,60], [68,58], [83,69], [87,76], [74,78], [71,71], [58,69], [54,62], [54,67], [37,84], [41,94], [2,99], [7,64], [22,60], [25,62], [18,54], [4,50], [13,40], [18,40], [24,42], [25,38], [41,26], [45,21], [44,35], [58,35], [62,32].

Genotypen, Phänotypen, Kodierung und Fitness: In der Literatur wurden speziell für das TSP verschiedene Kodierungen vorgeschlagen. Wir wollen die Städte einfach durchnummerieren (1 . . . 30). Damit ist ein ein Genotyp gegeben durch eine Sequenz von 30 Zahlen x_i ($i \in [1 \ldots 30]$), wobei $x_i \neq x_j$ für alle $i \neq j$ gilt (da jede Stadt nur einmal besucht werden soll). Zum Beispiel kodiert der Genotyp

```
8 30 7 4 ...  1 29
```

den Phänotyp "Beginne Reise in Stadt 8, fahre von 8 nach 30, von 30 nach 7, usw., und abschließend von 1 nach 29". Im Falle des TSP ist die Fitness eines Phänotyps gegeben durch die Länge der erforderlichen Reise. Als Abstand zwischen je zwei Städten verwenden wir einfach die euklidische Distanz, welche mittels der oben angegebenen Koordinaten leicht berechnet werden kann.

Populationsgröße *PopSize*: Gibt an, wieviele Genotypen in einer Population enthalten sind (und in diesem Sinn gleichzeitig betrachtet werden). Typische Grßen: PopSize $=$ 30, 50, 100, 500.

Selektion: Ausgehend von einer gegebenen Population wird ein bestimmter Prozentsatz (bezeichnet als $FractionSurvive$) der besten verfügbaren Genotypen zunächst unverändert in die neue Population übernommen (später werden diese Genotypen noch mit einer gewissen Wahrscheinlichkeit mutiert). Bei einer Populationsgröße von 50 und $FractionSurvive = 30$ werden also die 15 besten Genotypen ausgewählt. Der fehlende Anteil – also $(100 - FractionSurvive)$ Prozent – der Nachfolger-Population wird durch Rekombination generiert. Typische Werte für $FractionSurvive$: 30, 50 und 60 Prozent. Das Prinzip, Fitness-orientiert zu selektieren, wird auch mit dem Slogan "survival of the fittest" charakterisiert.

Rekombination: Zunächst werden 2 Elter, E1 und E2, abhängig von ihrer Fitness aus der Ausgangspopulation ausgewählt; je größer die Fitness, desto größer soll die Wahrscheinlichkeit sein, dass der entsprechende Genotyp ausgewählt wird. Nachdem E1 und E2 ausgewählt wurden, werden zufällig zwei sogenannte Schnittpunkte S1 und S2, mit $1 \leq$ S1, S2 ≤ 30, bestimmt. Das Kind von E1 und E2 wird nun generiert, indem von E1 der Bereich zwischen den Schnittpunkten unverändert übernommen wird und beginnend nach S2 die fehlenden Positionen durch Verwendung der entsprechenden Positionsbelegungen von E2 "aufgefüllt" (also belegt) werden. Bei diesem Auffüllen wird E2 zyklisch verwendet und stets darauf geachtet, dass im neuen Genotyp keine Stadt doppelt enthalten ist. Ein Beispiel macht dies klar (es seien 12 Städte zu besuchen und es sei S1 $=$ 6 und S2 $=$ 10):

```
E1:     8 11 3 5 6 4  -S1-  2 12 1  -S2-  9  7  10
E2:     1 2  3 4 5 6  -S1-  7 8  9  -S2-  10 11 12
Kind:   4 5  6 7 8 9        2 12 1        10 11 3
```

Zu beachten ist, dass das Kind also wieder einen gültigen Lösungsvorschlag kodiert (jede Stadt kommt genau einmal vor). Aus je zwei Vorgängern wird also ein Nachfolger generiert; dies wird so oft gemacht, bis die neue Population vollständig ist. Diese Rekombinationsform wird in der Literatur als *order crossover* bezeichnet.

Mutation: Mittels Mutation werden "punktuelle", also kleine Änderungen "im Erbmaterial", also in den Genotypen, vorgenommen. In dieser Übung wird angenommen, dass jeder neue Genotyp – also sowohl die entsprechend ihrer Fitness direkt ausgewählten als auch die durch Rekombination entstandenen – mit der Wahrscheinlichkeit $Mutate$ mutiert werden. (Selektion und Rekombination liefern also eine "vorläufige Nachfolger-Population", die dann durch Mutation in die endgültige Nachfolger-Population transformiert wird.) Die Mutation selbst bestehe einfach im Vertauschen der Werte (Städte) zweier zufällig gewählter Positionen im Genotyp.

Falls zum Beispiel zufällig die Positionen 1 und 3 gewählt wurden, dann ergibt sich folgendes (es seien wieder 12 Städte zu besuchen):

```
Genotyp:     8 11 3 5 6 4  2 12 1  9  7  10
             |     |
mutiert:     3 11 8 5 6 4  2 12 1  9  7  10
```

Durch diese Mutation, die in der Literatur auch als *SWAP Operator* bezeichnet wird, erhält man wieder einen gültigen Lösungsvorschlag. Typische Werte für $Mutate$ sind: 5, 10, 15 Prozent.
Neue Population: Diese liegt vor, nachdem die direkte Fitness-orientierte Selektion, die Rekombination und die Mutation durchgeführt wurden. Die Generierung wird solange fortgesetzt, bis entweder eine geeignete Lösung gefunden wurde (eine, die "gut genug" ist) oder bis eine maximale Anzahl ($MaxPop$) von Populationen erzeugt wurde. Beispielwerte für $MaxPop$: 30, 50, 100, 500.

OPTIONAL – 4-1-1. Wie oben beschrieben, werden die bei einer Rekombination verwendeten Elterteile E1 und E2 abhängig von Ihrer Fitness ausgewählt. Beschreiben Sie knapp verbal (und bei Bedarf unter Verwendung von Formeln), wie Sie diese Fitness-orientierte Auswahl programmiertechnisch realisiert haben. (Sollten Sie nicht in der Lage sein, Fitness-orientierte Auswahl zu realisieren, dann führen Sie die nachfolgenden Experimente mit "zufälliger Auswahl" durch.)

OPTIONAL – 4-1-2. Überlegen Sie sich alternative Rekombinationsverfahren. Generieren die von Ihnen vorgeschlagenen Verfahren nur gültige Lösungsvorschläge?

4-1-3. Implementieren Sie, unter Berücksichtigung der oben aufgeführen Parameter, den evolutionären Algorithmus. Wenden Sie Ihre Implementierung auf das oben angegebene 30-Städte-Problem an, wobei Sie die Parameter – $PopSize, MaxPop, FractionSurvive, Mutate$ – möglichst systematisch variieren. Sie können sich dabei an die oben angegebenen "typischen Werte" orientieren. Listen Sie, für die von Ihnen verwendeten Parameterkonstellationen, den jeweils besten Genotyp samt seines Fitnesswertes auf. Welche Feststellungen (oder Vermutungen) lassen sich an Hand Ihrer experimentellen Resultate treffen?

Bearbeitungen in PDF- *und* Source-Format (World, LaTeX, etc.)
bis zur nächsten Vorlesung an: gerhard.weiss@maastrichtuniversity.nl

---

EVOLUTIONÄRER ALGORITHMUS

1. Zufällige Initialisierung einer Menge M von Genotypen, also zufällige Generierung einer Anfangspopulation.

2. Initialisierung einer maximalen Anzahl von zu bildenden Populationen ($MaxPop$).

3. Initialisierung eines Zählers $AktPop$ ("aktuelle Population") und Belegung mit dem Wert 1.

4. Übergang von den in M enthaltenen Genotypen zu den Phänotypen und Bewertung (Bestimmung der Fitness) der Phänotypen. (Je nach Codierung und Optimierungsproblem kann dieser Schritt sehr einfach oder sehr komplex sein.)

5. Abbruch, falls ein geeigneter Phänotyp gefunden wurde; Ausgabe der Lösung.

6. Solange kein geeigneter Phänotyp gefunden wurde und $AktPop < MaxPop$:

   6.1. Bildung einer neuen Population N durch Anwendung von Selektion, Rekombination und Mutation auf die Genotypen in M.

   6.2. $M \longleftarrow N$

   6.3. Übergang von den in M enthaltenen Genotypen zu den Phänotypen und Bewertung der Phänotypen.

7. Ausgabe der besten gefundenen Lösung.

ABBILDUNG 4-1: Allgemeines Schema des evolutionären Algorithmus.