Teoria algorytmów i obliczeń – Projekt – Etap 1

Błażej Bobko, Jakub Gocławski, Patryk Kujawski, Radosław Kutkowski

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska

1 Opis zadania

Zadanie polega na znalezieniu deterministycznego automatu skończonego, będącego rekonstrukcją struktury automatu na podstawie relacji indukowanej przez język. Na wejściu otrzymujemy automat, który pozwala sprawdzić czy: $\forall_{x,y\in\Sigma^*}\ xR_Ly$ i na podstawie odpowiedzi otrzymanych przez powyższą funkcję tworzymy automat wynikowy. Automat ten próbujemy znaleźć za pomocą kolejnych optymalizacji za pomocą algorytmu PSO.

2 Struktury danych

Rozdział ten opisuje najważniejsze struktury danych potrzebne do rozwiązania zadania. Przedstawione są kluczowe atrybuty klas, z pominięciem atrybutów prywatnych i tymczasowych.

Automaton

Automat reprezentujemy jako klasę posiadającą następujące atrybuty:

- $LettersCount$ – liczba liter w alfabecie, litery numerujemy od 0 do $LettersCount$ - 1,
- $StatesCount$ – liczba stanów automatu, stany numerujemy od 0 do $StatesCount$ - 1,
- $CurrentState$ – indeks stanu, w którym aktualnie znajduje się automat,
- $Transistions$ – tablica opisująca funkcję przejścia; jest to tablica dwuwymiarowa, rozmiaru ($StatesCount$ * $LettersCount$) o wartosciach $[0\ ..\ StatesCount-1]$.

Przy tak zdefiniowanej strukturze danych, obliczenie automatu dla litery $Letter$ wyraża się następująco:

$$CurrentState = Transistions[CurrentState][Letter]$$

Particle

- $Position$ – aktualny stan cząsteczki, na podstawie którego zbudować można funkcje przejścia dla automatu; tablica rozmiaru ($StatesCount$ * $LettersCount$) o wartościach typu zmiennopozycyjnego z zakresu $[0.0\ ..\ StatesCount-1]$,
- $Error$ – wartość błędu dla aktualnej pozycji $Position$,
- $BestError$ – najlepszy „lokalny” błąd uzyskany dotychczas przez te cząsteczkę,
- $BestPosition$ – najlepszy „lokalny” stan cząsteczki, tablica zdefiniowana tak samo jak $Position$,
- $Velocity$ – aktualna prędkość cząsteczki; tablica rozmiaru ($StatesCount$ * $LettersCount$) o wartościach typu zmiennopozycyjnego z zakresu $[-(StatesCount-1)\ ..\ StatesCount-1]$.

PSO

Klasa realizująca optymalizację za pomocą algorytmu PSO. Atrybuty:

- $ParticleCount$ – liczba cząsteczek w roju,
- $Swarm$ – tablica rozmiaru $ParticleCount$ z wartościami typu $Particle$,
- $BestGlobalPosition$ – najlepszy „globalny” stan spośród wszystkich cząsteczek; tablica zdefiniowana tak samo jak $Particle.Position$,
- $BestGlobalError$ – najlepszy (najniższy) „globalny” błąd spośród wszystkich cząsteczek.

Dodatkowe parametry określające charakterystykę działania optymalizacji (np. parametr $maxEpochs$) przekazywane są bezpośrednio do funkcji uruchamiającej obliczenia PSO.

DataSet

- $Word1$ – pierwsze słowo (lista liter),
- $Word2$ – drugie słowo (lista liter),
- $AreRelated$ – wartość boolowska określająca, czy $Word1$ oraz $Word2$ są ze sobą w relacji.

3 PSO - Particle Swarm Optimization

PSO (*Particle Swarm Optimization*) jest metodą obliczeniową polegającą na iteracyjnych próbach ulepszenia rozwiązania problemu optymalizacji poprzez przeszukiwanie przestrzeni rozwiązań w wielu różnych punktach jednocześnie. Potencjalne rozwiązania, nazwane dalej cząsteczkami, są porównywane za pomocą pewnej funkcji dopasowania a następnie "przesuwają się" w przestrzeni rozwiązań. Nowa pozycja jest dobierana na podstawie kilku czynników m.in. najlepszego znanego globalnie rozwiązania oraz najlepszego rozwiązania unikalnego dla danej cząstki. Dokładny algorytm wygląda następująco:

Dane wejściowe

- N – ilość cząsteczek
- Dim – wymiar przestrzeni przeszukiwania
- $LowerBound$ – wektor długości Dim zawierający w i-tej komórce największą wartość współrzędnej możliwą w i-tym wymiarze przestrzeni rozwiązań
- $UpperBound$ – wektor długości Dim zawierający w i-tej komórce największą wartość współrzędnej możliwą w i-tym wymiarze w przestrzeni rozwiązań
- $MaxEpochs$ – Liczba iteracji po których zwracamy najlepsze znalezione rozwiązanie
- $VelocityWeight$ – Waga jaką cząsteczka przywiązuje dla swojej aktualnej prędkości
- $CognitiveWeight$ – Waga jaką cząsteczka przywiązuje do najlepszego rozwiązania jakie napotkała
- $SocialWeight$ – Waga jaką cząsteczka przywiązuje do najlepszego rozwiązania jakie znaleziono w toku działania PSO
- $Fitness$ – Funkcja $R^{Dim} \rightarrow R$, zwracająca dla danego rozwiązania pewną wielkość opisującą to jak bardzo bliskie optymalnemu jest to rozwiązanie. Wartość tej funkcji chcemy minimalizować/maksymalizować.

Sposób działania PSO przedstawiony jest za pomocą Algorytmu 1 na stronie 3.

4 Zbiór treningowy i testowy

Utworzenie zbiorów treningowego i testowego nastąpi poprzez wygenerowanie odpowiednio dużych zbiorów słów. Zbiór treningowy zawierać będzie wszystkie pary słów krótkich (długości 5) nad danym alfabetem i losowe pary słów długich. Słowa długie zostaną wygenerowane w sposób losowy oraz poprzez (wielokrotną) konkatenację słów krótkich, a następnie w sposób losowy dobrane w pary. Zbiór testowy będzie zawierał wszystkie wygenerowane pary słów długich, które nie weszły w skład zbioru testowego. Zbiory treningowe i testowe zostaną wygenerowane raz i zapisane w plikach tekstowych.

Algorytm 1 PSO

```
1: procedure PSO
2:     BestSocialValue ← ∞
3:     stwórz N cząsteczek
4:     for i := 0 to N do
5:         stwórz wektor Location, w taki sposób, że
6:         for j := 0 to Dim do
7:             Location_i(j) ← Random(LowerBound(j), UpperBound(j))
8:         end for
9:         BestCognitive_i ← Location_i
10:        BestCognitiveValue_i ← Fitness(Location_i)
11:        if BestCognitiveValue_i < BestSocialValue then
12:            BestSocialValue ← BestCognitiveValue_i
13:            BestSocial ← BestCognitive_i
14:        end if
15:        ustal wektor Velocity_i w losowy sposób, tak, że
16:        for j := 0 to Dim do
17:            size ← |UpperBound(j) − LowerBound(j)|
18:            Velocity_i(j) ← Random(−size, size)
19:        end for
20:    end for
21:    for k := 0 to MaxEpochs do
22:        for i := 0 to N do
23:            for j := 0 to Dim do
24:                RandomSocial = Random(0, 1)
25:                RandomCognitive = Random(0, 1)
26:                Velocity_i(j) ← VelocityWeight ∗ Velocity_i(j) + SocialWeight ∗ RandomSocial ∗
   (BestSocial(j) − Location_i(j)) + CognitiveWeight ∗ RandomCognitive ∗ (BestCognitive_i(j) −
   Location_i(j))
27:                Location_i(j) ← Location_i(j) + Velocity_i(j)
28:            end for
29:            fitness ← oblicz wartoć Fitness(Location_i)
30:            if fitness < BestCognitiveValue_i then
31:                BestCognitive_i ← Location_i
32:                BestCognitiveValue_i ← fitness
33:                if BestCognitiveValue_i < BestSocialValue then
34:                    BestSocial ← BestCognitive_i
35:                    BestSocialValue ← BestCognitiveValue_i
36:                end if
37:            end if
38:        end for
39:    end for
40:    return BestSocial
41: end procedure
```

5 Opis rozwiązania

Dane wejściowe

- A – automat wejściowy, podlegający rekonstrukcji
- M – liczba stanów automatu
- Al – wielkość alfabetu
- $MaxEpochs$ – liczba iteracji od ostatniego zaktualizowania najlepszego kandydata po których kończymy działanie algorytmu
- $MaxState$ – maksymalna liczba stanów jaką chcemy rozpatrzyć
- $LastBestCount$ – liczba cząsteczek stworzonych na podstawie najlepszego rozwiązania z poprzedniej iteracji (dla $LastBestCount = 0$ rozwiązania poprzedniej iteracji nie będą brane pod uwagę w inicjalizowaniu PSO w iteracji następnej)
- $DeathProbablility$ – szansa na zniszczenie cząsteczki i zastąpienie jej inną, losową (dla $DeathProbability = 0$ funkcjonalność umierania cząsteczek jest wyłączona)
- P – zbiór parametrów wywołania PSO

Działanie

Dla każdej pary słów w zbiorze par treningowych wywoływane jest działanie automatu A. Wyniki zapisywane są w słowniku, gdzie kluczem jest para sprawdzanych słów, a wartością wyniki obliczeń automatu A. Słownik ten posłuży do sprawdzania współczynników błędu automatów konstruowanych w toku działania algorytmu PSO.

Wywołujemy PSO dla każdego całkowitego i z przedziału $[2, MaxState]$ oraz ilości cząstek równej $P.N + LastBestCount$. Zmienna i określa ilość stanów jaką ma mieć automat wynikowy. Aby uprościć działanie funkcji $Fitness$ współrzędne cząstek zapisujemy w postaci macierzy $M * Al$ liczb zmiennopozycyjnych zamiast wektora o długości $M * Al$. Funkcja ta dla danej macierzy m zaokrągla wszystkie wartości w komórkach tej macierzy w dół do liczby całkowitej i konstruuje automat skończony opisany w sekcji 2 („Struktury danych"). Następnie dla każdej pary słów w zbiorze treningowych par sprawdzane jest czy oba słowa kończą działanie automatu w tym samym stanie. Następnie na podstawie wyników przetwarzania wstępnego sprawdzamy, czy wynik ten jest błędem. Zwracana jest liczba napotkanych błędów. W każdej iteracji prócz pierwszej do zbioru losowych cząstek dodawane są cząstki konstruowane na podstawie wyniku poprzedniej iteracji. Są to macierze z dodatkowym wierszem odpowiadającym nieosiągalnemu przez automat stanowi, wiersz ten wypełniany jest losowo, tak samo początkowy wektor prędkości cząsteczki również jest losowy. W każdym kroku algorytmu PSO każda cząsteczka ma szansę „zginąć", w takim wypadku nadawany jej jest losową prędkość i losowa pozycja w przestrzeni rozwiązań.

Prawdopodobieństwo to opisane jest zmienną $DeathProbability$. Rozwiązanie to ma na celu minimalizację wpływu minimów lokalnych na proces przeszukiwania przestrzeni.

Zmianie uległy również warunki zakończenia działania instancji PSO. PSO zwraca najlepszą wartość nie po danej ilości kroków, a po $MaxEpochs$ kroków od ostatniego zaktualizowania najlepszego kandydata na rozwiązanie. Nie spowoduje to nieskończonych obliczeń przy asymptotycznej zbieżności kolejnych kandydatów do optymalnego rozwiązania ze względu na to, że funkcja Fitness przyjmuje wartości naturalne.