Использование эволюционного подхода в задаче подбора заданной строки символов

Спасибо hoarfrost'у и Ричарду Докинзу
за задачу и идею решения!

Постановка задачи.

Дана целевая строка длины n символов.
Генотип – произвольная строка той же длины.
Функция приспособленности f – число отличий фенотипа от целевой строки.

Особенности задачи.

1. Быстрое вырождение популяции (можно обойтись популяцией малого объема):
   - для популяции из 2 особей на нахождение решения требуется $[33\,000, 75\,000]$ итераций;
   - из 10 особей – $[35\,000, 56\,000]$;
   - из 100 особей – $[31\,000, 61\,000]$.
2. Сильная зависимость от мутаций:
   - 2 особи – $[35\,000, 75\,000]_{\alpha=0,1} \rightarrow [8\,100, 38\,000]_{\alpha=0,2} \rightarrow [6\,600, 16\,000]_{\alpha=0,5}$;
   - 10 особей – $[35\,000, 56\,000]_{\alpha=0,1} \rightarrow [11\,000, 33\,000]_{\alpha=0,2} \rightarrow [7\,000, 11\,000]_{\alpha=0,5}$;
   - 100 особей – $[35\,000, 56\,000]_{\alpha=0,1} \rightarrow [6\,000, 14\,000]_{\alpha=0,5}$.
3. Наличие у целевой функции только одного экстремума.
4. Невозможность получения некорректного решения (нет ограничений).

Стратегия решения.

1. На каждой итерации выбираются 2 наиболее приспособленные особи A и B по критерию $f_i \pm rd \rightarrow \min$, где $r \in [0,1]$ – случайное число с равномерным распределением, d – степень разброса (эмпирически выбрано $d = 0{,}1$).
2. Между ними проводится скрещивание с целью получения потомка C по принципу $C_i = \begin{cases} A_i, r > 0{,}5 \\ B_i, r < 0{,}5 \end{cases}$ – равновероятный выбор генов родителей.
3. Определяется, необходимо ли проводить мутацию по критерию $\alpha r_i > r_{i+1}$, где α – вероятность мутации. Если мутация выполняется, то случайная позиция строки потомка замещается на случайный символ алфавита. Например, «A**B**C» $\rightarrow$ «A**Z**C» – символ во второй позиции (выбрана случайно) заменен с «B» на «Z» (выбран случайно).
4. Потомок замещает наименее приспособленную особь популяции по критерию $f_i \pm rd \rightarrow \max$.
5. Итерации повторяются до тех пор, пока вся популяция не соберется в глобальном экстремуме.

Листинг программы.

```
program Strs;
{$APPTYPE CONSOLE}

uses
  SysUtils;

const
  EthalonStr: String = 'TWO BEER OR NOT TWO BEER';          { Эталонная строка }
  POPULATION_SIZE = 10;                                     { Размер популяции }
  ALLOWED_CHARS: String = 'QWERTYUIOPASDFGHJKLZXCVBNM ';    { Алфавит }
```

```
  MUTATION_PROBABILITY = 0.1;                              { Вероятность мутации }

var
  PopulationArr: array [1..POPULATION_SIZE] of String;
  AvgFitness: Double;


{ Случайная инициализация популяции }
procedure RandomInit();
var
  I, StrPos, J: Integer;
begin
  for I := 1 to POPULATION_SIZE do begin
    SetLength(PopulationArr[I], Length(EthalonStr));

    for J := 1 to Length(PopulationArr[I]) do begin
      StrPos := Random(Length(ALLOWED_CHARS)) + 1;
      PopulationArr[I][J] := ALLOWED_CHARS[StrPos];
    end;
  end;
end;


{ Приспособленность Index-ного представителя популяции }
function Fitness(Index: Integer): Integer;
var
  I: Integer;
begin
  Result := 0;
  for I := 1 to Length(EthalonStr) do
    Result := Result + Byte(EthalonStr[I] <> PopulationArr[Index][I]);
end;


{ Вывод популяции на экран, попутно расчет средней приспособленности }
procedure WritePopulation();
var
  I, Sum, CurrFitness: Integer;
begin
  Sum := 0;

  for I := 1 to POPULATION_SIZE do begin
    CurrFitness := Fitness(I);
    Sum := Sum + CurrFitness;

    Writeln(I:2, ' ', PopulationArr[I], ' (', CurrFitness, ')');
  end;

  AvgFitness := Sum/POPULATION_SIZE;

  Writeln('Average fitness = ', AvgFitness:2:2);
end;


{ Скрещивание }
function Crossover(MotherIndex, FatherIndex: Integer): String;
var
  I: Integer;
begin
  SetLength(Result, Length(EthalonStr));
  for I := 1 to Length(EthalonStr) do
    if Random > 0.5 then
      Result[I] := PopulationArr[MotherIndex][I]
    else
      Result[I] := PopulationArr[FatherIndex][I]
end;


{ Мутация }
procedure Mutation(var Str: String);
var
  Pos, CharPos: Integer;
begin
  Pos := Random(Length(EthalonStr)) + 1;            { Случайная позиция в строке }
  CharPos := Random(Length(ALLOWED_CHARS)) + 1;     { Случайный символ }

  Str[Pos] := ALLOWED_CHARS[CharPos];
end;
```

```
{ Расчет приспособленности (с отклонением \pm rd) }
function FitnessWithDeviation(SourceFitness: Integer): Double;
const
  DEVIATION = 5;
begin
  Result := SourceFitness * (1 + Random*DEVIATION - DEVIATION/2);
end;


{ Выбор наиболее приспособленного родителя (SkipThis - можно пропустить уже выбранного ранее) }
function GetBest(SkipThis: Integer = -1): Integer;
var
  BestFitness, CurrFitness: Double;
  BestIndex, I: Integer;
begin
  { Выбираем случайного родителя (пропорционально приспособленности) }
  BestFitness := FitnessWithDeviation( Fitness(1) );
  BestIndex := 1;

  for I := 2 to POPULATION_SIZE do begin
    if I = SkipThis then
      continue;

    CurrFitness := FitnessWithDeviation( Fitness(I) );

    if CurrFitness < BestFitness then begin
      BestFitness := CurrFitness;
      BestIndex := I;
    end;
  end;

  Result := BestIndex;
end;


{ Выбор наименее приспособленной особи }
function GetWorst(): Integer;
var
  WorstFitness, CurrFitness: Double;
  WorstIndex, I: Integer;
begin
  { Выбираем случайную наименее приспособленную особь (пропорционально приспособленности) }
  WorstFitness := FitnessWithDeviation( Fitness(1) );
  WorstIndex := 1;

  for I := 2 to POPULATION_SIZE do begin
    CurrFitness := FitnessWithDeviation( Fitness(I) );

    if CurrFitness > WorstFitness then begin
      WorstFitness := CurrFitness;
      WorstIndex := I;
    end;
  end;

  Result := WorstIndex;
end;


var
  S: String;
  MotherIndex, FatherIndex, DeadIndex, Iteration: Integer;

begin
  RandSeed := 0;
  RandomInit();

  Iteration := 0;

  repeat
    Writeln('Iteration = ', Iteration);
    WritePopulation();
    Writeln;

    { Выбор особей для скрещивания }
    MotherIndex := GetBest();
    FatherIndex := GetBest(MotherIndex);

    { Скрещивание }
    S := Crossover(MotherIndex, FatherIndex);
```

```
    { Мутация }
    if Random*MUTATION_PROBABILITY > Random then
      Mutation(S);

    { Выбор наиболее слабой особи }
    DeadIndex := GetWorst();

    { Новая особь заменяет наименее приспособленную }
    PopulationArr[DeadIndex] := S;

    Inc(Iteration);

    { Пришли в глобальный максимум? }
    if AvgFitness = 0.0 then
      break;
  until False;

  { Вывод итоговой популяции }
  Writeln('Iteration = ', Iteration);
  WritePopulation();
  Writeln;

  Writeln('Done');
  Readln;
end.
```

Пример.

Целевая строка «TWO BEER OR NOT TWO BEER».

Протокол эволюции.

```
Iteration = 0                          ; Номер итерации при моделировании
 1 TTHGINGYKZWDNMUSTAGTSYKH (21)       ; Номер_особи Фенотип (Приспособленность)
 2  QPREIVXMZCHDHCJTMEBLVTU (23)
Average fitness = 22.00

Iteration = 1
 1 TTHGINGYKZWDNMUSTAGTSYKH (21)
 2 TTHGINGYKZWDNMUSTAGTSYKH (21)
Average fitness = 21.00

[skip]

Iteration = 961
 1 TTFCPIAOPSQDNMUWTDGTGIKH (21)
 2 TTFCPIAOPSQDNMUWTWGTGIKH (20)
Average fitness = 20.50

Iteration = 962
 1 TTFCPIAOPSQDNMUWTDGTGIKH (21)
 2 TTFCPIAOPSQDNMUWTWGTGIKH (20)
Average fitness = 20.50

Iteration = 963
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWGTGIKH (20)
Average fitness = 20.00

[skip]

Iteration = 1091
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.50

Iteration = 1092
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.50

Iteration = 1093
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.50

Iteration = 1094
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.50

Iteration = 1095
 1 TTFCPIAOPSQDNMUWTWGTGIKH (20)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.50

Iteration = 1096
 1 TTFCPIAOPSQDNMUWTWOTGIKH (19)
 2 TTFCPIAOPSQDNMUWTWOTGIKH (19)
Average fitness = 19.00
```

[skip]
УМЕР ПРИСПОСОБЛЕННЫЙ ПРЕДОК, ОТКАТ НА ШАГ НАЗАД В ЭВОЛЮЦИИ

Iteration = 2329
 1 LFBPNAHGUQGNZEQTWOT THC (20)
 2 TLJBPNAHGUQGNZEQTWOT THC (19)
Average fitness = 19.50

Iteration = 2330
 1 LFBPNAHGUQGNZEQTWOT THC (20)
 2 LJBPNAHGUQGNZEQTWOT THC (20)
Average fitness = 20.00

[skip]

Iteration = 2480
 1 LFBPNAHGUOGNZEQTWOT THC (20)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.50

Iteration = 2481
 1 LFBPNAHGUOGNZEQTWOT THC (20)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.50

Iteration = 2482
 1 LFBPNAHGUOGNZEQTWOT THC (20)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.50

Iteration = 2483
 1 LFBPNAHGUOGNZEQTWOT THC (20)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.50

Iteration = 2484
 1 LFBPNAHGUOGNZEQTWOT THC (20)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.50

Iteration = 2485
 1 LFBPNAHGUOGNZTQTWOT THC (19)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.00

[skip]
БЫЛА "ПЛОХАЯ МУТАЦИЯ", НО МЕНЕЕ ПРИСПОСОБЛЕННЫЙ ПОТОМОК НЕ ПРИЖИЛСЯ

Iteration = 2514
 1 LFBPNAHGUOGNZTQTWOT THC (19)
 2 LFBPNAHGUOGNZTQTMOT THC (20)
Average fitness = 19.50

Iteration = 2515
 1 LFBPNAHGUOGNZTQTWOT THC (19)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 19.00

[skip]

Iteration = 2521
 1 LFBBNAHGUOGNZTQTWOT THC (18)
 2 LFBPNAHGUOGNZTQTWOT THC (19)
Average fitness = 18.50

Iteration = 2522
 1 LFBBNAHGUOGNZTQTWOT THC (18)
 2 LFBBNAHGUOGNZTQTWOT THC (18)
Average fitness = 18.00

[skip]

Iteration = 11145
 1 IOOITUWN OB TOHETJOCVEVR (15)
 2 IOOITUWN OB TOHETJOCVBVR (16)
Average fitness = 15.50

Iteration = 11146
 1 IOOITUWN OB TOHETJOCVEVR (15)
 2 IOOITUWN OB TOHETJOCVEVR (15)
Average fitness = 15.00

[skip]

Iteration = 23194
 1 AWO BSVT UR FTTETYOKBEEW (11)
 2 AWO BSVT UR FOTETYOKBEEW (10)
Average fitness = 10.50

[skip]

Iteration = 34082
 1 TWO BZEF OR MOT TKOLBEEY (6)
 2 TWO BEEF OR MOT TKOLBEEY (5)
Average fitness = 5.50

Iteration = 34083
 1 TWO BEEF OR MOT TKOLBEEY (5)
 2 TWO BEEF OR MOT TKOLBEEY (5)
Average fitness = 5.00

[skip]

Iteration = 63844
 1 TWO BEER OR NOT TNO BEER (1)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.00

Iteration = 63845
 1 TWO BEER OR NOT TNO BEER (1)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.00

Iteration = 63846
 1 TWO BEER OR NWT TNO BEER (2)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.50

Iteration = 63847
 1 TWO BEER OR NWT TNO BEER (2)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.50

Iteration = 63848
 1 TWO BEER OR NWT TNO BEER (2)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.50

Iteration = 63849
 1 TWO BEER OR NWT TNO BEER (2)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 1.50

Iteration = 63850
 1 TWO BEER OR XWT TNO BEER (3)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 2.00

Iteration = 63851
 1 TWO BEER OR NOT TWO BEER (0)
 2 TWO BEER OR NOT TNO BEER (1)
Average fitness = 0.50

Iteration = 63852
 1 TWO BEER OR NOT TWO BEER (0)
 2 TWO BEER OR NOT TWO BEER (0)
Average fitness = 0.00