ZADÁNÍ SEMESTRÁLNÍ PRÁCE

OPTIMALIZACE FUNKCE GENETICKÝM ALGORITMEM

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která bude hledat **extrém funkce pomocí genetického algoritmu**.

Genetický algoritmus je technika, která napodobuje procesy známe z biologie a hledá tak optimální řešení zadaného problému. Podle zadaného problému se nadefinuje tzv. *fitness funkce*, jejíž hodnota vyjadřuje životaschopnost potomka – vhodnost řešení problému.

Základní princip genetického algoritmu:

1. Náhodně vytvoříme několik různých řešení zadané úlohy – jedinců.
2. Provedeme křížení jedinců.
3. Náhodně zmutujeme malý počet jedinců.
4. Ověříme, jak dobře jeden každý jedinec řeší náš problém tak, že pro každého získáme hodnotu fitness funkce.
5. Vytvoříme novou generaci.
6. Opakujeme body 2 až 5.

Pro účely této semestrální práce je pro lepší pochopení problematiky naprosto dostačující článek o genetickém algoritmu na Wikipedii. Pro detailní zkoumání genetických algoritmů doporučuji pročíst tutoriál Genetic Algorithms[2]. Pro lepší představu o široké škále problémů, které je možno touto cestou řešit viz Užitečné odkazy.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Parametry programu

Aplikace bude přijímat z příkazové řádky 2 povinné argumenty:

- jméno souboru, ve kterém jsou metadata o testované funkci,
- počet generací, které mají být provedeny.

Porgram bude přijímat také jeden nepovinný argument:

- **-m** *procento mutací (0-100).*

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 9.1 Stretch s jádrem verze 3.2.78-1 x86_64 a s překladačem gcc 6.3.0.

[2] `https://www.tutorialspoint.com/genetic_algorithms`

Váš program tedy může být během testování spuštěn například takto:

```
...\>gms.exe e:\functions\meta.txt 1000 -m 5
```

Pokud nebudou na příkazové řádce uvedeny alespoň dva argumenty, vypište chybové hlášení a stručný návod k použití programu (v angličtině).

Popis programu

Optimalizovaná funkce

Funkce, kterou budete optimalizovat, bude dodána ve formě programu ve spustitelné podobě. Tento program bude načítat konfiguraci ze souboru s metadaty, který je popsán níže. Vaše práce se tedy bude skládat z několikanásobných úprav tohoto souboru a následném spuštění externího programu, který pro vás hodnotu vypočte. Za svého běhu může na standardní výstup vypisovat podrobné informace o svém běhu, na poslední řádku vypíše hodnotu funkce, kterou optimalizujete, a ukončí se. Jméno souboru s metadaty bude předáno jako parametr vašemu programu.

Formát souboru s metadaty

Soubor na prvním řádku obsahuje příkaz pro spuštění programu, který simuluje funkci, uvozený znaky "#_". Následuje výčet všech proměnných, které tento program načte pro výpočet hodnoty funkce. Pro hledání optima laďte pouze proměnné, které mají na předchozí řádce uvedený definiční obor a za ním typ proměnné (reálné číslo – `R` / celé číslo – `Z`) striktně ve formátu popsaném dále, ostatní ponechte beze změn.

```
#_(a,b);$
jmeno=hodnota
```

kde *(a,b)* je definiční obor a *$* je typ proměnné (R/Z).

Tedy v následujícím konfiguračním souboru budete ladit pouze proměnou **X**. Proměnnou **Z** ponecháte beze změn:

```
#_e:\functions\fce.exe

#_(5,90);Z
X=0
Z=25
```

Mapování genu

Pro mapování genu se doporučuje použít dále uvedený způsob[3].

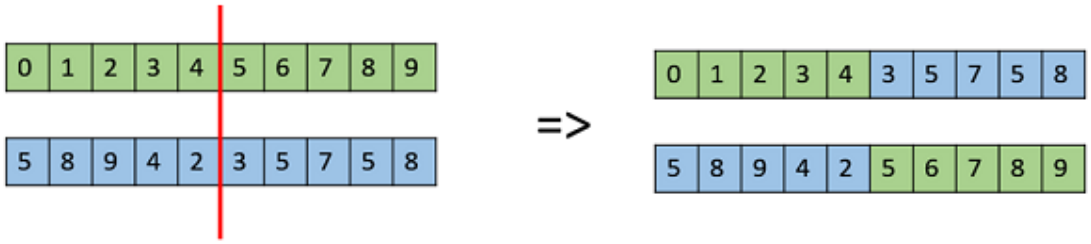
- Pro celá čísla použijte jejich binární zápis.
- Pro reálná čísla použijte konkrétní hodnoty daných proměnných.

[3] Pokud byste však rádi vyzkoušeli jinou variantu a přinesli nový vlastní nápad, přijďte jej zkonzultovat a společně vyhodnotíme, zda je takový alternativní přístup vhodný.

Křížení

Křížení dvou potomků poté realizujte pro každou skupinu genů[4] takto:

- Pro binární zápis celého čísla kombinujte části genů rodičů (viz obr. 1). Hranici bloků vybírejte vždy náhodně, ale tak, aby spadla mezi bity kódující interval uvedený u proměnné.
- Pro hodnotu reálného čísla vypočítejte pro potomka novou hodnotu jako aritmetický průměr hodnot rodičů.



Obrázek 1: Ukázka tvorby nových potomků zeleného a modrého rodiče
Zdroj: www.tutorialspoint.com

Mutace

Mutaci genotypu provádějte následujícím způsobem. Podle parametru z příkazové řádky **-m** vyberte určitou část z populace[5], kterým náhodně upravíte část genů. Část binární reprezentace celého čísla **náhodně přepíšete** nulami a jedničkami. U reálného čísla reprezentovaného hodnotou **vygenerujete náhodně nové číslo** z uvedeného intervalu. U mutovaného potomka však vždy upravte **jen jednu skupinu genů**.

Dejte však pozor, ať mutací neponičíte své nejlepší jedince.

Výstup

Výsledkem činnosti programu budou dva textové soubory (**gen.txt** a **val.txt**), ve kterých bude souhrn průběhu hledání optima. Protože celý proces může být časově velice náročný, tyto soubory tvořte průběžně.

Po každé generaci na začátek souboru **gen.txt** připište nejlepšího jedince (konkrétní hodnoty optimalizovaných proměnných a výsledek funkce). Výsledný soubor tedy bude řazen tak, že první v souboru jsou nejnovější výsledky.

Po každém získání hodnoty optimalizované funkce zapište do souboru **val.txt** konkrétní hodnoty laděných proměnných a výsledek funkce.

Ukázka výstupu v gen.txt

[4] Soubor genů popisující jednu vlastnost – v našem případě je to jedna konkrétní proměnná mapovaná binárně nebo hodnotou.

[5] Pokud není použit přepínač -m zvolte hodnotu 5 %.

```
--- GENERATION 598 ---
58.236
X=45#(5,90);Z
Y=0.28#(0.1,1.0);R

--- GENERATION 597 ---
57.786
X=40#(5,90);Z
Y=0.24#(0.1,1.0);R
.
.
```

Ukázka výstupu val.txt

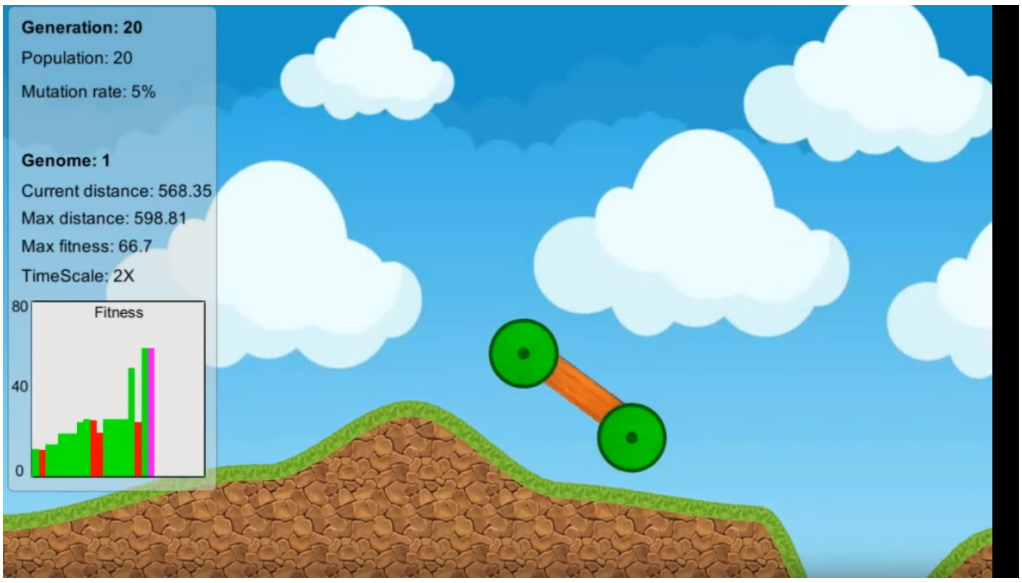
```
58.236
X=7#(5,90);Z
Y=0.11#(0.1,1.0);R

58.236
X=9#(5,90);Z
Y=0.15#(0.1,1.0);R

.
.
.
```

Užitečné techniky a odkazy

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové techniky, které podle vás nejlépe povedou k cíli.

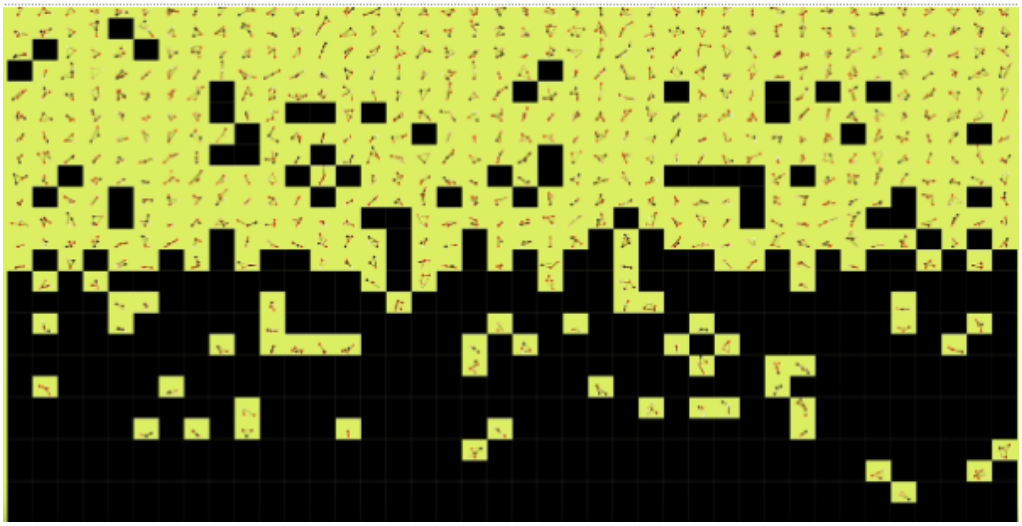
Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. Genetické algoritmy
   `https://www.tutorialspoint.com/genetic_algorithms`
2. Genetic algorithm – popis techniky na Wikipedii,
   `https://en.wikipedia.org/wiki/Genetic_algorithm`
3. Genetic algorithms – evolution of a 2D car in Unity
   `https://www.youtube.com/watch?v=FKbarpAlBkw`



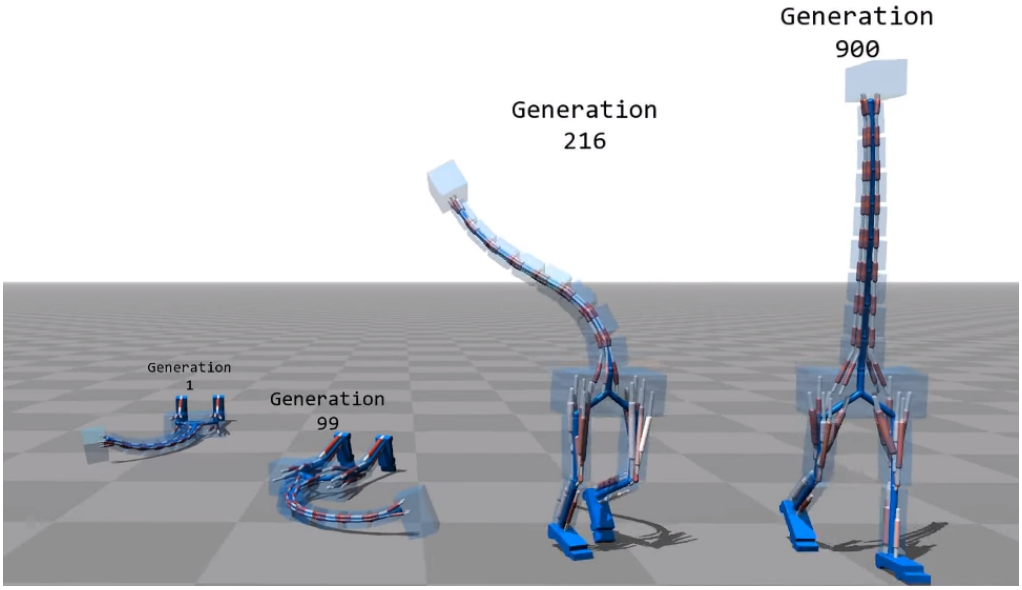
Obr. 2.: Genetic algorithms – evolution of a 2D car in Unity

4. Evolution Simulator
   `https://www.youtube.com/watch?v=GOFws_hhZs8`



Obr. 3.: Evolution Simulator

5. Flexible Muscle-Based Locomotion
   `https://www.youtube.com/watch?v=pgaEE27nsQw`



Obr. 4.: Evolution Simulator