Projekt do předmětu SFC
Demonstrace činnosti PSO v jazyce C++

Vendula Poncová (xponco00)

6. prosince 2015

1 Úvod

V této zprávě se věnuji optimalizační technice Particle Swarm Optimization a její implementaci. Popisuji návrh demonstrační aplikace, realizaci a uživatelské rozhraní. Nakonec uvádím experimenty, které jsem provedla s aplikací, a jejich výsledky.

2 Optimalizace hejnem částic

Particle Swarm Optimization neboli optimalizace hejnem částic je optimalizační technika inspirovaná přírodou. Je založená na simulaci pohybu ptačího hejna hledajícího potravu. Místo ptáků však uvažujeme částice a místo potravy řešení optimalizačního problému. Každá částice pak má nějakou pozici, pohybuje se danou rychlostí, pamatuje si místo s nejlepší dosaženou hodnotou účelové funkce a je si vědoma nejlepší dosažené hodnoty účelové funkce hejna. Krok optimalizačního algoritmu pak spočívá v přesunu každé částice na pozici danou vektorem rychlosti, výpočtem nové rychlosti a určením nové nejlepší dosažené hodnoty částice i hejna.

Nechť t je čas, dt je časový interval pro jeden krok, d je dimenze a n je počet částic. Nechť $k \in \{1, 2, ..., n\}$ je pořadové číslo částice. Pak x^k je poloha částice, v^k je vektor rychlosti, x^k_{best} je nejlepší nalezená pozice částice a x_{best} je nejlepší nalezená pozice hejna. Hodnota p^k je hodnota účelové funkce $f(x^k)$. Výpočet pozice definován následovně:

$$x^k(t + dt) = x^k(t) + v^k(t) \cdot dt$$

Nechť ω je koeficient setrvačnosti, c_p je kognitivní váhový koeficient, c_g je sociální váhový koeficient, r_p a r_g jsou náhodné hodnoty z intervalu $(0, 1)$, pak rychlost v^k lze spočítat vztahem:

$$v^k(t + dt) = \omega v^k(t) + c_p r_p(x^k_{best}(t) - x^k(t)) + c_g r_g(x_{best} - x^k(t))$$

Algoritmus probíhá následovně:

1. Náhodně nastavte počáteční pozice a rychlosti částic.
2. Nastavte nejlepší dosažené řešení každé částice na současnou pozici a hodnotu účelové funkce.
3. Určete nejlepší dosaženou hodnotu a pozici hejna.
4. Pro každou částici vypočítejte novou pozici a rychlost částice. Aktualizujte nejlepší dosažené řešení částice a hejna.
5. Pokud se řešení nezlepšuje nebo byl překročen čas výpočtu, výpočet ukončete, jinak opakujte předchozí krok.

3 Aplikace

Cílem projektu bylo vytvořit v jazyce C/C++ aplikaci, která bude sloužit k demonstračním účelům. V následujících kapitolách popisuji návrh této aplikace, její implementaci, sestavení, spuštění a způsob ovládání uživatelského rozhraní.

3.1 Návrh aplikace

Při návrhu jsem kladla důraz na vysvětlení principu optimalizace hejnem částic. Chtěla jsem především znázornit vliv jednotlivých faktorů na výslednou rychlost.

3.2 Popis implementace

V souboru `lib.h` jsou definované pomocné funkce. Třída `Point` ze souboru `point.h` reprezentuje pozici v dvourozměrném prostoru. V souboru `function.h` jsou deklarované účelové funkce.

Částice je reprezentovaná třídou `Particle` ze souboru `particle.h`. Objekt v sobě nese informaci o aktuální poloze částice (`position`), rychlosti (`velocity`), hodnotě účelové funkce pro aktuální i nejlepší dosaženou pozici (`value, bestValue, bestPosition`). Pro potřeby vykreslování částice objekt dále obsahuje jednotlivé vektory, ze kterých se skládá vektor rychlosti (`vw, vp, vg`) a souřadnice následující pozice (`nextPosition`).

Optimalizaci reprezentuje třída `Optimization` v souboru `optimization.h`. Třída v sobě nese parametry optimalizace, seznam částic a funkce pro manipulaci s částicemi. Funkce `addParticle` přidá částici na danou pozici, `moveParticle` částici posune a pomocí funkcí `computeNextPosition`, `computeNextVelocity`, `computeFitness`, `updateLocalBest` a `updateGlobalBest` aktualizuje stav částice a hejna. Funkce `moveParticleSwarm` pak spustí algoritmus optimalizace a vypočítá optimální hodnotu.

`View` je třída, která v sobě informace o stavu zobrazení: přiblížení, pozice středu, výška a šířka plátna, ukazatel na označenou částici, ... Umožňuje tak přepočítávat

Aplikace se vytvoří a spustí ve funkci `main` souboru `main.cpp`. Hlavní okno aplikace je reprezentované třídou `MainWindow` ze souboru `mainwindow.h`. Zde se rovněž propojí prvky uživatelského prostředí s řídícím objektem třídy `Driver` ze souboru `driver.h`. Třída `Driver` přijímá a zpracovává uživatelské akce, nastavuje objekty typu `View` a `Optimization` a upravuje grafické uživatelské prostředí. V souboru `canvas.h` je deklarovaná plocha pro vykreslení grafu funkce a částic. V souborech `tabconfig.h` a `tabparticle.h` jsou deklarované třídy pro panely Configuration a Particle.

3.3 Sestavení a spuštění

Aplikace vyžaduje pro sestavení a spuštění knihovnu Qt v minimální verzi 4.6. O instalaci knihovny bohatě informují stránky `www.qt.io`. Na serveru merlin a v CVT na operačním systému CentOS lze aplikaci spustit příkazem ./pso. V CVT na operačním systému Windows 7 je třeba spustit soubor `pso_win7/bin/pso.exe`.

Na serveru merlin lze překlad provést příkazem `qmake; make` z adresáře se zdrojovými soubory src. Aplikaci lze spustit příkazem `./pso`. Na operačním systému Windows je třeba otevřít program Qt Creator, v nabídce File vybrat možnost Open File or Project a otevřít soubor `pso_win7/src/pso.pro`. Překlad lze spustit v nabídce Build možností Build All. Možnost Run spustí aplikaci.

Z toho důvodu, že spustitelné soubory byly příliš velké a nebylo možné je odevzdat v archivu xponco00.zip, lze na stránce

`www.dropbox.com/sh/5ydb41uzbwn25f3/AABIX6InhYvPsWtR_RTaeII2a?dl=0`

stáhnout kompletní archiv xponco00-all.zip, jehož součástí jsou zdrojové i spustitelné soubory pro Windows.

3.4 Ovládání uživatelského rozhraní

Po spuštění aplikace se otevře okno s panelem nástrojů, konfiguračním panelem Configuration, informačním panelem Particle a plochou s vykresleným grafem účelové funkce. Optimalizaci lze nastavit v konfiguračním panelu. Význam jednotlivých položek je následující:

Fitness function Účelová funkce f.

Cognitive factor Kognitivní váhový koeficient c_p.

Social factor Sociální váhový koeficient c_g.

Inertia factor Koeficient setrvačnosti ω.

Max initial velocity Hodnota maximální počáteční rychlosti v_{max}.

Current iteration Počet provedených kroků.

Max iterations Maximální počet kroků.

Best fitness Nejlepší dosažená hodnota účelové funkce.

Best solution Nejlepší dosažená pozice.

Set default values Nastaví se přednastavené hodnoty.

Kliknutím na plochu grafu lze vytvořit částici. Bude umístěna na danou souřadnici x a nulovou souřadnici y. Pokud umístíte kurzor na vytvořenou částici, zobrazí se v panelu Particle informace o nejlepším dosaženém řešení dané částice, pozici a rychlosti. Dále se v grafu vykreslí vektor rychlosti (velocity) a jeho jednotlivé složky (social velocity v_g, cognitive velocity v_p, inert velocity v_ω). Zároveň se vykreslí nejlepší dosažená pozice vybrané částice (best) a její následující pozice (next). Kliknutím na částici se tyto informace budou zobrazovat trvale. Lze tak snadněji sledovat pohyb částice v průběhu optimalizace. Opětovným kliknutím na částici informace zmizí. Po celou dobu je též vykreslovaná nejlepší dosažená pozice hejna (modrý neoznačený čtvereček). Kolečkem myši se lze v grafu přibližovat a oddalovat. Průběh optimalizace lze ovládat pomocí panelu nástrojů. Lze provádět následující akce:

Random Na náhodných pozicích se vygeneruje 10 částic.

Clear Všechny vytvořené částice se smažou.

Step Provede se jeden krok optimalizačního algoritmu.

Run Spustí se animace kroků optimalizačního algoritmu.

Stop Animace se zastaví.

Finish Zobrazí se stav po dosažení maximálního počtu kroků.

Restart Všechny částice se smažou, zapomene se nejlepší řešení a restartuje se čítač kroků.

4 Experimenty

Správnost implementace lze ověřit sledováním pohybu částice a jednotlivých složek rychlostí. Následující pozice částice leží vždy ve směru vektoru rychlosti. Vzhledem k tomu, že časový krok dt je nastaven na 1, tak velikost vektoru rychlosti odpovídá vzdálenosti mezi současnou a následující pozicí vektoru. Provedením jednoho kroku optimalizace se částice přesune na následující pozici. Výpočet pozice tedy probíhá v pořádku.

Vektor rychlosti v se počítá jako součet tří vektorů v_ω, v_p, v_g. Vektor v_ω je ovlivněn předchozím vektorem rychlosti a koeficientem setrvačnosti ω. Při provádění kroků optimalizace si lze všimnout, že směr vektoru v_ω se mění se zpožděním jednoho kroku na směr vektoru rychlosti v. Lze též pozorovat zvětšení nebo zmenšení vektoru v_ω v závislosti na hodnotě *omega*. Vektor v_g vždy směřuje na nejlepší nalezenou pozici hejna a jeho velikosti se mění v závislosti na p_g. Vektor v_p pak směřuje na nejlepší nalezenou pozici částice a jeho velikost se mění v závislosti na p_g. Vektor v vždy odpovídá součtu vektorů v_ω, v_p a v_g. Výpočet rychlosti tedy také probíhá v pořádku.

Dále jsem ověřovala, jaké je nalezené nejlepší řešení pro danou účelovou funkci. Pro funkci $sin(x)$ a 10 náhodných částic jsem nalezla maximální hodnotu 1. U funkce $-abs(x)$ jsem pro 10, 20, 30, 40 i 50 částic nalezla hodnotu blízkou 0, ale nikdy 0. U Easomovy funkce jsem nalezla optimální hodnotu 1 už pro 10 částic. Pro Holderovu funkci bylo často nejlepší dosaženou hodnotou účelové funkce lokální maximum. U Ackleyho funkce byla hodnota vždy blízká 0, tedy globálnímu maximu. Pro Weierstrassovu funkci byly nalezeny hodnoty kolem 0.393751.

5 Závěr

Navrhla jsem a implementovala aplikaci pro demonstraci optimalizační techniky Particle Swarm Optimization. Implementaci jsem provedla v jazyce C++ a s knihovnou Qt. Správnost implementace jsem ověřila provedenými experimenty.