VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Implementace ACO s grafickým rozšířením
Projekt SFC

Bc. Jan Holáň, xholan11@stud.fit.vutbr.cz

1. prosince 2024

Obsah

1 Úvod

Optimalizace mravenčími algoritmy (ACO), je využívána v grafových úlohách, zejména v mnoha oblastech logistiky, ale i dalších odvětvích. Je inspirována kolektivním chováním mravenců, kteří hledají potravu a pomocí feromonu, který vypouštějí, dávají dalším jedincům z mravenčí populace zprávu o tom, kolik potravy našli. Toto chování se obecně využívá pro hledání nejkratší trasy, a proto všechny problémy, které se pomocí mravenčích algoritmů řeší, musí být na problém nejkratší trasy převedeny.

V této práci je popsána implementace řešení problému obchodního cestujícího (TSP) pomocí mravenčího algoritmu Ant Colony System (ACS), který vychází z původního algoritmu Ant System. Dále je zde ukázána práce s uživatelským rozhraním, které bylo vytvořeno pro tento projekt a pro názornou ukázku práce tohoto algoritmu.

2 TSP

Problém obchodního cestujícího (angl. Travelling Salesman Problem, zkr. TSP), je velmi známý pro svou časovou složitost (problém spadá do třídy NP). Řešením je nalezení nejkratšího průchodu grafem, přičemž každý vrchol (město) je navštíven pouze jednou. Důležité je poznamenat, že obchodník, cestující městy, musí začít i skončit ve stejném městě (což je reflektováno v implementaci).

Tento problém byl také zvolen pro demonstraci algoritmu ACS. Protože se u všech úloh a definic problému TSP počítá s možností přejití mezi libovolnými dvěma body v grafu, je i v této implementaci vyžadováno, aby byl zadaný graf **úplný**.

3 ACS

[1] Ant Colony System (ACS), mající svůj základ v algoritmu Ant System (AS), který je implementací základní podoby ACO, se řídí následujícím postupem:

Algorithm 1 Algoritmus ACS

while není dosaženo počtu iterací **do**

získej trasu pro každého mravence;

aktualizuj na trasách (globálně) množství feromonu;

pokud bylo nalezeno lepší řešení než doposud, ulož jej;

end while

Mravenci v algoritmu ACS se dále řídí pravděpodobnostním výběrem hrany v grafu mezi vrcholy r, s pro mravence k, přičemž mravenec se nachází ve vrcholu r. Vzorec 1 obsahuje následující symboly:

- η - vliv dané hrany. Obvykle vypočítána jako $\frac{1}{len(hrana)}$
- τ - vliv feromonu na dané hraně
- α - váha parametru τ (v některých článcích bývá α opomenuta, v tomto projektu je nicméně použita)
- β - váha parametru η

$$p_k(r,s) = \frac{(\tau_{r,s}^{\alpha})(\eta_{r,s}^{\beta})}{\sum_{dostupne(y)} (\tau_{r,y}^{\alpha})(\eta_{r,y}^{\beta})} \tag{1}$$

Mezi další použité parametry patří (pod danými symboly figurují v dále popisované implementaci):

- Q - číslo regulující přírůstek feromonu podle jeho globální aktualizace, viz 3.2,
- ρ - číslo mezi 0 a 1, udávající míru vypařování feromonu, hraje roli v jeho lokální aktualizaci (3.1),

- n - počet mravenců pro spuštění algoritmu,
- $q0$ - **číslo mezi 0 a 1 zvyšující šanci, že si mravenec vybere hranu s nejnižší váhou** - jedná se o důležitý rys algoritmu **ACS**,
- τ_0 - počáteční hodnota feromonu na hranách (zadaná buď jako číslo, nebo jako způsob, viz dále 3.3, 5.3,
- $alpha\_decay$ - číslo mezi 0 a 1 figurující ve výpočtu globální aktualizace feromonu (na podobné bázi jako ρ u lokální aktualizace).

3.1 Lokální aktualizace feromonů

Při průchodu grafem jednotlivými mravenci je feromon aktualizován lokálně, což spočívá ve vypařování feromonu, který byl na hraně před příchodem mravence, a jeho částečném doplnění. Při tom je zajištěno, aby feromon neklesl pod jeho počáteční úroveň, která je nastavena při inicializaci grafu.

3.2 Globální aktualizace feromonů

Po tom, co všichni mravenci najdou své průchody grafem (v tomto případě i suboptimální řešení TSP), dojde k aktualizaci feromonů na všech cestách tak, že cesty, které navštívil **nejlepší** mravenec, feromon získají, zatímco u všech ostatních dojde pouze k jeho vypaření. Toto je specifikum ACS.

3.3 Inicializace feromonů

Při vytvoření grafu je všem hranám přiřazena počáteční úroveň feromonu. Ta je buď nastavena na nějaké konkrétní číslo (např. 0.001), nebo pomocí nějakého vzorce, přičemž nejčastěji se používá vzorec

$$\tau_0 = \frac{1}{N * greedy\_solution}$$

kde *greedy_solution* je délka cesty v grafu, nalezena greedy algoritmem a vyhovující suboptimálnímu řešení TSP, a N je počet uzlů v grafu.

4 Implementace knihovny pro řešení TSP pomocí ACS

Kód je implementován jako knihovna, nacházející se v adresáři *src/acs*, jejíž hlavní prvky jsou ve třídách `ACOWorld`, který zodpovídá za vytvoření grafu se zadaným souborem uzlů a volitelně se zadaným souborem hran. Pokud hrany nejsou zadané, pak jsou vytvořeny mezi všemi uzly s váhami rovny hodnotám euklidovy vzdálenosti mezi danými uzly. V této třídě se nachází i funkce pro inicializaci feromonu na jednotlivých hranách - ta je volaná ze třídy `ACOSolver`. Pokud graf není úplný, vrací třída chybu v podobě *Exception*.

Třída `ACOSolver` řídí celý průběh hledání nejkratší trasy v grafu, začínající i končící ve stejném uzlu (který je volitelně zadán). Hlavní funkce `solve()` provádí zadaný počet iterací. V nich je vždy volána funkce `__do_ants_solutions()`, kde probíhá spolupráce se třídou `ACOAnt`, která implementuje pohyb mravence v grafu podle algoritmu ACS. Dalšími dvěma důležitými funkcemi jsou `__local_update_pheromones()` a `__global_update_pheromones()`, které se chovají podle popisů výše (3.1, resp.3.2).

5 Spuštění, GUI a instalace

5.1 Instalace

Celý projekt je napsán v jazyce **Python 3.12.7**. Z knihoven byly použity *numpy* a *pyqt5*. Instalace potřebných závislostí je provedena skriptem *install.sh*, umístěným v kořenovém adresáři projektu. Je vyžadováno administrátorské oprávnění - pro spuštění nutno upravit heslo administrátora.

5.2 Formát vstupních souborů

V této části kapitoly je popsán požadovaný formát vstupních souborů. Program vyžaduje zadání uzlů, hrany nejsou povinné.

Uzly grafu musí být předány programu v textovém souboru, kde na každém novém řádku se musí vyskytovat jeden uzel ve formátu: *id_uzlu;název_uzlu;souřadnice_x;souřadnice_y*. Soubor může obsahovat volné řádky. Umožněny jsou také řádky s komentářem, kde takový řádek musí začínat znakem #.

Soubor má stejné podmínky jako soubor s uzly, pouze formát hran se liší: *id_prvního_uzlu;id_druhého_uzlu;váha_hrany*.

5.2.1 Data

V adresáři *data* lze najít několik ukázkových souborů uzlů (případně kompatibilních hran) a složku *tests*, kde se nachází vstupní soubory, na kterých byly testovány různé okrajové případy.

5.3 Příkazová řádka

Skript s názvem *acstsp.py* (adresář *src*) slouží pro spouštění algoritmu z příkazové řádky. Jeho parametry jsou následující (lze nechat vypsat pouhým spuštěním skriptu bez argumentů):

```
[-h] [--edge_file EDGE_FILE] [--alpha ALPHA] [--beta BETA]
[--rho RHO] [--n N] [--tau0 TAU0] [--Q Q] [--q0 Q0]
[--alpha_decay ALPHA_DECAY] [--start_node START_NODE]
[--iterations ITERATIONS] [--verbose] [--display]
node_file
```

Jediný povinný argument je cesta k souboru s uzly grafu. Argument `tau0` může navíc nabýt hodnoty "greedy". Argument `verbose` zajistí okomentovaný běh programu a `display` zobrazí graf s výslednou cestou na konci běhu algoritmu. Všechny ostatní argumenty mají číselnou povahu.

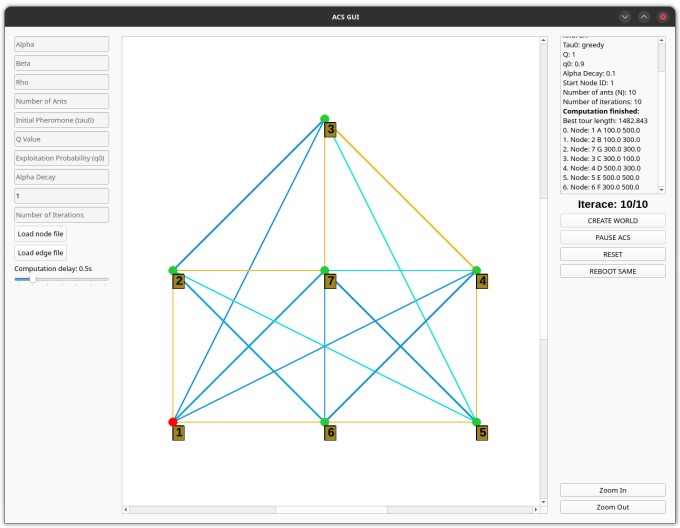
5.3.1 Ukázkové spuštění příkazové řádky s výstupem

Pozn.: předpokládané spuštění terminálu v *kořenové* složce projektu.

```
$ python3 src/acstsp.py data/input1_classic_nodes.in
--edge_file=data/input1_classic_edges.in --n=20
--iterations=20

--- Algorithm Parameters ---
node_file: data/input1_classic_nodes.in
edge_file: data/input1_classic_edges.in
[...next parameters...]
----------------------------

Best tour cost:  135.0
****** Best tour edges ******
1 --> 2 | 5.0 0.020049099466349474
2 --> 4 | 20.0 0.03467690688779829
3 --> 4 | 60.0 0.020049099466349474
1 --> 3 | 50.0 0.03467690688779829
****** Best tour nodes ******
1 OSTRAVA 100.0 300.0
2 ZLIN 50.0 60.0
4 OPAVA -200.0 30.0
3 BRNO 100.0 -80.0
```



Obrázek 1: GUI pro ovládání ACS řešící problém TSP

5.4 GUI

Grafické rozhraní pro spouštění algoritmu ACS za účelem řešení problému TSP je spustitelné z příkazové řádky (očekáváme terminál spuštěný v kořenovém adresáři projektu) příkazem `$ python3 src/acstspgui.py`. Jeho implementace se nachází v adresáři *src/gui*. Výsledkem spuštění je běh programu s GUI (na obrázku 1), dokud není *okno* programu zavřeno.

V rámci grafického rozhraní je možné

- nahrát soubory s uzly a volitelně s hranami (tlačítka **Load {node,edge} file**),
- nastavovat libovolně parametry pro algoritmus ACS (textová pole v levé části okna),
- nechat si vykreslit nahraný graf (tlačítko **CREATE WORLD**),
- ovládat výpočet a jeho rychlost (a tím i stav animace) (*slider* vlevo, tlačítko **START ACS**),
- nechat resetovat původní graf (tlačítko **REBOOT SAME**),
- resetovat celé rozhraní (tlačítko **RESET**),
- a oddalovat a přibližovat plochu (tlačítka **Zoom {In, Out}**).

Výsledky lze potom pozorovat v hlavním okně, kdy barevnost hran ukazuje koncentraci feromonu na dané hraně v barevné škále *aqua* až *modrá*. Výsledná nalezená optimální cesta v grafu je vybarvena *žlutě*. Konkrétní data o výsledcích a parametrech algoritmu jsou vypisována do pravého horního okna.

5.5 Spouštěcí skript

Skript pod názvem *tryscript.sh*, nacházející se v kořenovém adresáři projektu, spustí dva běhy algoritmu skrz rozhraní příkazové řádky, výstupy budou vypsány do terminálu. Výsledek druhého běhu by se měl zobrazit v dialogovém okně.

Reference

[1] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53–66, 1997. Available: `https://api.semanticscholar.org/CorpusID:14074696`