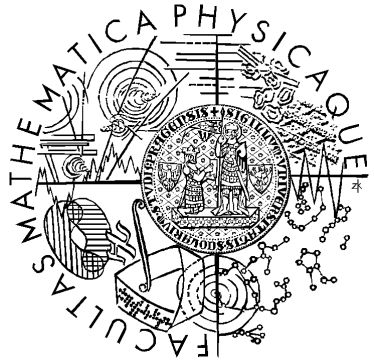
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Martin Jureček

Biologicky motivované algoritmy a možnosti jejich využití v kryptoanalýze

Katedra algebry

Vedoucí bakalářské práce: Doc. RNDr. Jiří Tůma, DrSc.
Studijní program: Matematika
Matematické metody informační bezpečnosti

2009

Ďakujem svojmu vedúcemu Doc. RNDr. Jiřímu Tůmovi, DrSc. za cenné rady a pripomienky k práci.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 7.8.2009 Martin Jureček

Obsah

Názov práce: Biologicky motivované algoritmy a možnosti jejich využití v kryptoanalýze
Autor: Martin Jureček
Katedra (ústav): Katedra algebry
Vedúci bakalárskej práce: Doc. RNDr. Jiří Tůma, DrSc.
e-mail vedúceho: Jiri.Tuma@mff.cuni.cz

Abstrakt: V predloženej práci študujeme vybrané biologicky motivované algoritmy Particle Swarm Optimization a Ant Colony Optimization, ktoré sú založené na sociálnom správaní jedincov v spoločenstve. Tieto algoritmy sa používajú pri riešení optimalizačných úloh a dosahujú veľmi dobré výsledky. V tejto práci je popísané použitie vybraného algoritmu Particle Swarm Optimization na riešenie NP-úplného problému obchodného cestujúceho a je porovnaný s jedným z klasických algoritmov, Metódou rezných nadrovín. Ďalšou témou práce je použitie biologicky motivovaných algoritmov pre kryptoanalýzu. Uvádzame kryptoanalýzu jednoduchej zámeny pomocou metódy Particle Swarm Optimization.

Kľúčové slová: Particle Swarm Optimization, Ant Colony Optimization, Problém obchodného cestujúceho

Title: Biologically motivated algorithms and possibilities for their applications in cryptanalysis
Author: Martin Jureček
Department: Department of Algebra
Supervisor: Doc. RNDr. Jiří Tůma, DrSc.
Supervisor's e-mail address: Jiri.Tuma@mff.cuni.cz

Abstract: In this work we study two biologically motivated algorithms Particle Swarm Optimization and Ant Colony Optimization, that are based on social behavior of individuals in community. These algorithms are used to solve optimization tasks with very good results. In this work usage of Particle Swarm Optimization algorithm for solving the NP-complete travelling salesman problem is described and is compared with classic algorithm, Cutting-plane method. Other subject of this work is usage of biologically motivated algorithms for cryptology. We introduce cryptanalysis of simple substitution by Particle Swarm Optimization method.

Keywords: Particle Swarm Optimization, Ant Colony Optimization, Travelling Salesman Problem

1 Úvod

Biologicky motivované algoritmy, ďalej BMA, sú algoritmy určené na riešenie optimalizačných problémov, ktoré využívajú princípy sociálneho správania zvierat. Medzi hlavných predstaviteľov BMA patria algoritmy Particle Swarm Optimization, skrátene PSO a Ant Colony Optimization, ďalej ACO. Tieto algoritmy sa používajú pre riešenie zložitých problémov, ktorých riešenie pomocou klasických optimalizačných metód, ako napríklad lineárne programovanie alebo rôzne numerické metódy, je časovo veľmi náročné. PSO a ACO na jednej strane nezaručujú nájdenie optimálneho riešenia úlohy, ale na druhej strane dosahujú uspokojivé výsledky v relatívne krátkom čase.

Algoritmus PSO je založený na tom, že sa každý jedinec neustále snaží vylepšovať svoju pozíciu v spoločenstve. Jedinec pri hľadaní novej pozície zohľadňuje svoj pôvodný smer a rýchlosť, svoju doposiaľ najlepšiu pozíciu a nakoniec pozíciu svojho najúspešnejšieho suseda. V algoritme ACO jedinci spolu nepriamo komunikujú pomocou pachovej stopy, ktorú zanechávajú na svojej ceste. Jedinec pri výbere z viacerých možností, ktorou cestou sa vydá, zohľadňuje veľkosť koncentrácie pachovej stopy, pričom si vyberá cestu s vyššou koncentráciou pachových stôp.

Táto práca obsahuje popis algoritmov PSO a ACO, ďalej tu nájdeme použitie PSO pre problém obchodného cestujúceho a nakoniec pomocou rovnakého algoritmu sa pokúsime rozanalyzovať jednoduchú zámenu.

Druhá kapitola sa venuje metóde PSO a obsahuje okrem popisu algoritmu i základné informácie o parametroch PSO a štruktúre priestoru populácie. Sú v nej uvedené aj niektoré varianty PSO, ktoré sa odlišujú v spôsobe výpočtu vektoru rýchlosti jedinca.

V tretej kapitole je popisovaná metóda ACO. Na úvod sú spomenuté informácie o chovaní živých mravcov a to akým spôsobom vedia nájsť pomocou pachovej stopy najkratšiu cestu medzi mraveniskom a miestom s potravou. Okrem popisu algoritmu sú spomenuté dve varianty ACO, ktorých odlišný spôsob počítania hodnôt pachových stôp spôsobí zvýšenie efektivity algoritmu.

Na probléme obchodného cestujúceho je v štvrtej kapitole porovnaný algoritmus PSO ako predstaviteľ BMA s jedným z klasických algoritmov pre riešenie tohto problému, Metódou rezných nadrovín.

Piata kapitola sa venuje kryptoanalýze jednoduchej zámeny pomocou metódy PSO, kde autor modifikoval algoritmus PSO a navrhol spôsob výpočtu rýchlosti jedinca.

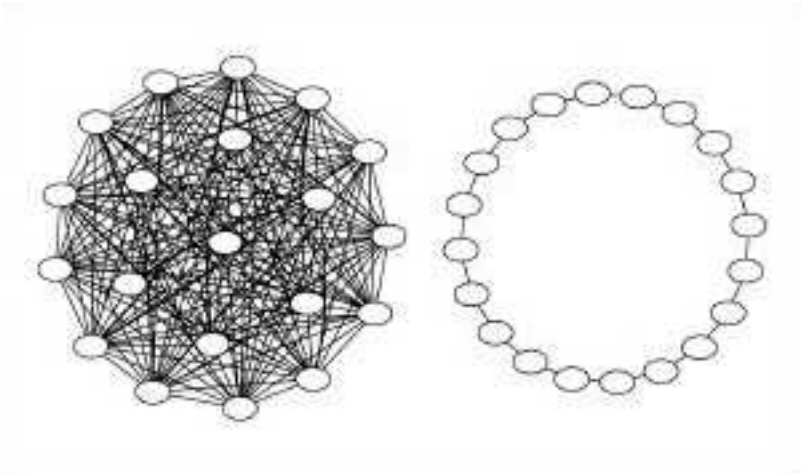
2 Particle Swarm Optimization

2.1 Úvod

Particle Swarm Optimization, ďalej PSO, je optimalizačná metóda motivovaná sociálnym správaním zvierat, ktorá slúži na hľadanie globálneho optima. PSO prvýkrát popísali v roku 1995 Dr. James Kennedy a Dr. Russell C. Eberhart. Táto metóda sa nechala inšpirovať napríklad kŕdľami vtákov alebo húfami rýb, ktorých charakteristikou je, že sa každý jedinec neustále snaží vylepšovať svoju pozíciu v spoločenstve. Podrobnejší prehľad o tejto metóde nájdeme v [5].

Jedinec je reprezentovaný ako bod v priestore, v ktorom sa pohybuje a za jeho susedov považujeme všetkých jedincov populácie, ktorí sú v určitom zmysle k nemu blízko. Podrobnejší popis susedstva je v podkapitole 2.4 s názvom Štruktúra populácie.

Každý jedinec komunikuje so všetkými svojimi susedmi a je ovplyvnený svojou doteraz najlepšou nájdenou pozíciou a najlepšou nájdenou pozíciou v systéme všetkých jedincov susedstva, kde doteraz nájdená najlepšia pozícia znamená, že do aktuálneho kroku algoritmu sa nenašla žiadna lepšia pozícia. Vhodnosť pozície udáva hodnota účelovej funkcie, tzv. *fitness funkcie*. Každý jedinec určuje jedno riešenie optimalizačného problému. Boli vypracované dve varianty PSO. Jedna s globálnym susedstvom a jedna s lokálnym susedstvom. Pre globálnu variantu má každý jedinec tendenciu sa blížiť k jeho doteraz najlepšej nájdenej pozícii a k najlepšej pozícii celého spoločenstva. Na druhú stranu pre lokálnu variantu sa každý jedinec blíži k jeho doteraz najlepšej pozícii a k najlepšej pozícii restringovaného susedstva. Vo variante s globálnym susedstvom sa priestor, v ktorom sa jedinci pohybujú, interpretuje ako úplný neorientovaný graf, aký je napríklad na obrázku 1 vľavo. Vo variante s lokálnym susedstvom sa často používa graf kružnica, na obrázku 1 vpravo.



Obrázok 1: Príklad najčastejšie používaných interpretácií priestoru

2.2 Popis algoritmu

Predpokladajme, že priestor, ktorý budeme prehľadávať je $n-$dimenzionálny euklidovský priestor $\mathbf{R}^n$. Pozíciu $i-$tého jedinca v spoločenstve reprezentujeme bodom $x_i \in \mathbf{R}^n$. Rýchlosť jedinca je reprezentovaná vektorom rýchlosti $\vec{v_i} \in \mathbf{R}^n$, ktorý jedincovi udáva smer a tiež vzdialenosť, ktorú má prejsť v príslušnom kroku algoritmu. Kvalitu pozície jedinca x_i udáva hodnota tzv. fitness funkcie f. Najlepšia pozícia je taká, ktorej hodnota fitness funkcie je najlepšia zo všetkých uvažovaných pozícií. Analogicky si môžeme definovať najhoršiu pozíciu. V prípade, že je optimalizačná úloha položená tak, že hľadáme minimálnu hodnotu fitness funkcie, tak za najlepšiu pozíciu považujme takú, ktorej hodnota fitness funkcie je najmenšia. Najlepšiu doteraz nájdenú pozíciu pre $i-$tého jedinca si označme $p_i \in \mathbf{R}^n$ a hodnotu fitness funkcie pre túto pozíciu si označme $pbest_i$. Doteraz najlepšiu nájdenu pozíciu v celom susedstve budeme značiť $p_g \in \mathbf{R}^n$ a hodnotu fitness funkcie príslušnú tejto pozícii označme $f(p_g)$. Teraz môžeme popísať algoritmus PSO. Základná kostra algoritmu je prevzatá z [5].

1. Inicializujme populáciu jedincov s náhodnými pozíciami x_i a náhodnými vektormi rýchlosti $\vec{v_i}$ v $n-$dimenzionálnom priestore, ktorý budeme prehľadávať.

2. Vonkajší cyklus. Ten končí najčastejšie v prípade dostatočne malého rozdielu hodnôt fitness funkcie v najhoršej pozícii a v najlepšej pozícii, alebo v prípade dosiahnutia maximálneho počtu iterácií algoritmu.

3. Vnútorný cyklus. Ten prebieha cez všetkých jedincov x_i v danej iterácii.

4. Ohodnoťme pozíciu jedinca x_i fitness funkciou f.

5. Porovnajme práve spočítanú hodnotu $f(x_i)$ s príslušnou hodnotou $pbest_i$. Ak je hodnota $f(x_i)$ lepšia ako hodnota $pbest_i$, tak do premennej $pbest_i$ priraďme $f(x_i)$ a do premennej p_i priraďme súčasnú pozíciu x_i.

6. Nájdime jedinca v susedstve s doposiaľ najlepšou hodnotou fitness funkcie a označme jeho pozíciu ako p_g.

7. Zmeňme rýchlosť a pozíciu $i-$tého jedinca pre nasledujúcu iteráciu takto:

$$\begin{aligned} \vec{v_i} &= \vec{v_i} + Rand(0, \phi_1) \otimes (p_i - x_i) + Rand(0, \phi_2) \otimes (p_g - x_i) \\ x_i &= x_i + \vec{v_i}. \end{aligned}$$

8. Koniec vnútorného cyklu

9. Koniec vonkajšieho cyklu.

Premenná $Rand(0, \phi_i)$ reprezentuje $n-$ticu náhodných čísel uniformne rozdelených v intervale $[0, \phi_i]$, ktorá je náhodne generovaná v každej iterácii a pre každého jedinca zvlášť. Operácia súčin $\otimes$ je definovaná nasledovne: Nech p a q sú body v $\mathbf{R}^n$, potom

$p \otimes q = (p_1.q_1, p_2.q_2, \ldots, p_n.q_n)$. Vzorec v 7. bode pre výpočet nového vektora rýchlosti pre nasledujúcu iteráciu vyjadruje tendenciu jedinca zostať vo svojom pôvodnom smere $\vec{v_i}$, blížiť sa k svojej doteraz najlepšej nájdenej pozícii p_i a blížiť sa k najúspešnejšiemu susedovi v spoločenstve p_g. Do akej mieri bude jedinec uvažovať pozície p_i a p_g určuje náhodná zložka $Rand(0, \phi_1)$, resp. $Rand(0, \phi_2)$.

2.3 Parametre PSO

Pre urýchlenie priebehu algoritmu a pre zvýšenie kvality výsledku je potrebné nájsť vhodné hodnoty parametrov. Základné parametre pre PSO sú: dimenzia priestoru, rozsah prehľadávaného priestoru, veľkosť populácie, obmedzenie zložiek vektora rýchlosti V_{max}, parametre ϕ_1 a ϕ_2 vyjadrujúce náhodnosť, ďalej maximálny počet iterácii a maximálna prípustná odchylka najlepšej a najhoršej hodnoty fitness funkcie. Niektoré hodnoty parametrov závisia na konkrétnom optimalizačnom probléme a nastavujeme ich empiricky, iné hodnoty parametrov sú dané.

Jeden z parametrov je dimenzia priestoru, v ktorom hľadáme optimálne riešenie a takýto parameter závisí len na danom optimalizačnom probléme, pričom dimenzia priestoru je určená počtom argumentov fitness funkcie. Aby sa jedince príliš od seba nevzdiaľovali, potrebujeme vymedziť rozsah prehľadávaného priestoru. Rovnako aj rozsah priestoru závisí na konkrétnom probléme.

Ďalej je potrebné nastaviť vhodnú veľkosť populácie, ktorá empiricky závisí na dimenzii priestoru a významne zvyšuje s rastom počtu jedincov výpočtovú zložitosť problému. Čím viac jedincov použijeme, tým hustejšie sa priestor prehľadá, ale výpočtová zložitosť bude väčšia. Najbežnejšia používaná veľkosť populácie je v rozmedzí 20 až 50 jedincov.

Rýchlosť jedincov má tendenciu rásť do vysokých hodnôt a tak by mohol jedinec opustiť priestor, v ktorom hľadáme optimum. Aby jedince nemali príliš veľkú rýchlosť, tak každá hodnota každej súradnice vektoru $\vec{v_i}$ je z intervalu $[-V_{max}, V_{max}]$. Približne v dobe august 2007 autori článku [5], ktorí významne prispeli k rozvoju PSO, nepoznali, ako nájsť optimálnu hodnotu V_{max}. Optimálna hodnota V_{max} znamená, že pre ňu algoritmus dosiahne najlepšie výsledky. Ak by bola príliš nízka, algoritmus môže predčasne konvergovať v lokálnom optime. Naopak ak by bola príliš vysoká, jedinec môže prekročiť hranice prehľadávaného priestoru. Po prekročení hranice priestoru bude jednotlivcovi vygenerovaná nová náhodná pozícia.

Parametre ϕ_1 a ϕ_2 používané v 7.bode algoritmu určujú, do akej miery sa náhodne zmenia príslušné vektory rýchlosti. Tieto parametre poskytujúce náhodnosť vyjadrujú tendenciu jedinca blížiť sa k svojej dočasne najlepšej pozícii alebo k pozícii najúspešnejšieho suseda. Veľká hodnota $Rand(0, \phi_1)$ podporuje jedinca vrátiť sa na svoju doteraz najlepšiu pozíciu a zase veľká hodnota $Rand(0, \phi_2)$ uprednostňuje posun jednotlivca k najlepšiemu jedincovi v celej populácii. Najčastejšie sa používajú hodnoty parametrov $\phi_1 = \phi_2 = 2$, hoci tieto hodnoty môžu spôsobovať divergenciu algoritmu, t.j. jedinci budú prekračovať hranice prehľadávaného priestoru. Nastavenie parametrov ϕ_1 a ϕ_2 súvisí s obmedzujúcim parametrom V_{max} zložiek vektora rýchlosti $\vec{v_i}$.

Ďalej je potrebné nastaviť parametre pre ukončenie vonkajšieho cyklu. Najčastejšie

vyjadrujú maximálny počet iterácií algoritmu a maximálnu prípustnú odchylku najlepšej a najhoršej hodnoty fitness funkcie. Tieto parametre závisia na zadanom probléme a na požiadavkách na kvalitu riešenia.

Aby sme mali lepšiu kontrolu nad pohybom jedincov v prehľadávanom priestor a aby sa čo najmenej prechádzali hranice priestoru, potrebujeme znížiť dôležitosť parametru V_{max}. Tak zavedieme parameter ω, tzv. zotrvačnosť. Rovnica pre výpočet rýchlosti sa nám upraví do nasledovného tvaru:

$$\vec{v_i} \quad = \quad \omega.\vec{v_i} + Rand(0,\phi_1) \otimes (p_i - x_i) + Rand(0,\phi_2) \otimes (p_g - x_i)$$

Parameter ω sa zvyčajne volí z intervalu $(0,1)$, pričom v úvodných iteráciách má vyššiu hodnotu, aby sa na začiatku prehľadal čo najväčší priestor a postupne sa hodnota parametra ω znižuje, čo spôsobí detailnejšie prehľadávanie okolia svojej dočasne najlepšej pozície, resp. pozície najlepšieho suseda v spoločenstve. Preto parameter ω závisí na parametroch ϕ_1 a ϕ_2.

Namiesto parametru zotrvačnosti ω môžeme používať parameter χ, ktorý má na vektor rýchlosti jedinca podobný vplyv ako ω. Hodnota χ opäť závisí na parametroch ϕ_1 a ϕ_2 a ak označíme $\phi = \phi_1 + \phi_2$, potom $\chi = 2/(\phi - 2 + \sqrt{\phi^2 - 4\phi})$, pričom tento výpočet parametru χ navrhli autori článku [5]. V tomto prípade sa nám vzorec pre výpočet rýchlosti prevedie do tvaru:

$$\vec{v_i} \quad = \quad \chi.(\vec{v_i} + Rand(0,\phi_1) \otimes (p_i - x_i) + Rand(0,\phi_2) \otimes (p_g - x_i))$$

V článku [5] sa ako najbežnejšia hodnota pre ϕ uvádza 4.1 a potom constriction coefficient χ výjde približne 0.7298. Takýmto spôsobom vieme eliminovať parameter V_{max} a zamedziť divergencii algoritmu.

Pre úplnosť spomeňme trochu odlišnú variantu PSO, nazývanu Fully Informed Particle Swarm, skrátene FIPS. Až doteraz bol každý jedinec ovplyvnený jeho doteraz najlepším výsledkom a najlepším výsledkom jedinca v susedstve, pričom informáciu o úspešnosti ostatných jedincov jeho susedstva sme neuvažovali. Vo verzii FIPS je každý jedinec ovplyvnený všetkými jeho susedmi, niekedy bez vplyvu na jeho predchádzajúce úspechy. Výpočet $\vec{v_i}$ potom vykonáme takto:

$$\vec{v_i} = \chi(\vec{v_i} + \frac{1}{K_i}.\sum_{n=1}^{K_i} Rand(0,\phi) \otimes (p_{nbr_n} - x_i)),$$

kde K_i je počet susedov $i-$tého jedinca a p_{nbr_n} je doposiaľ najlepšia nájdená pozícia $n-$tého suseda pre $i-$tého jedinca. S dobrými parametrami FIPS dosahuje lepšie výsledky ako štandardná metóda PSO, ale vzrastá závislosť na štruktúre priestoru populácie.

2.4 Štruktúra populácie

Pre populáciu jedincov umiestnenú do $n-$dimenzionálneho euklidovského priestoru $\mathbf{R}^n$ si množinu susedov nejakého jedinca, ktorý je na pozícii $x_i \in \mathbf{R}^n$, môžeme predstaviť

ako všetkých jedincov, ktorý sa nachádzajú v množine $\{y_i \in \mathbf{R}^n; \rho(x_i, y_i) \leq r\}$, kde ρ je euklidovská metrika, ktorá vyjadruje vzdialenosť medzi dvomi bodmi a r je vopred dohodnutá maximálna vzdialenosť od bodu x_i. V prípade, že populácia jedincov je umiestnená do diskrétneho priestoru, je definícia susedstva závislá na štruktúre daného diskrétneho priestoru. Štruktúra susedstva môže byť statická, t.j. taká, kde sa susedia ani susedstvá nemenia počas behu programu. Na druhej strane dynamická štruktúra sa v priebehu programu mení. Pretože pre variantu PSO s lokálnym susedstvom platí, že konverguje pomalšie a je menej náchylná na priťahovanie jedincov do lokálneho optima ako varianta PSO s globálnym susedstvom, tak môže byť vhodné začať s lokálnou variantou, kde sa uvažujú iba jedince z množiny $\{y_i \in \mathbf{R}^n; \rho(x_i, y_i) \leq r\}$, pre daný polomer r a pomaly zväčšovať veľkosť susedstva, až dospejeme k variante s globálnym susedstvom, kde každé dva jedince sú susedia.

Pre výpočet vektoru rýchlosti $\vec{v_i}$ a pozície x_i pre nasledujúcu iteráciu je výhodný $n-$dimenzionálny euklidovský priestor $\mathbf{R}^n$. V prípade diskrétneho priestoru môžu nastať problémy s interpretáciou vzorca $\vec{v_i} = \vec{v_i} + Rand(0, \phi_1) \otimes (p_i - x_i) + Rand(0, \phi_2) \otimes (p_g - x_i)$. V takom prípade môžeme vykonať výpočet v priestore $\mathbf{R}^n$ a vhodnou transformáciou previesť hodnoty do pôvodného diskrétneho priestoru. Táto transformácia by mala každému bodu $x_i = (x_{i,1}, \ldots, x_{i,n}) \in \mathbf{R}^n$ priradiť práve jeden bod z diskrétneho priestoru a pritom by mala zohľadniť relácie medzi zložkami bodu x_i.

2.5 Zhodnotenie PSO

Poznamenajme, že napriek tomu, že metóda PSO nám nezaručuje nájdenie optimálneho riešenia daného problému, podáva prijateľné riešenia v pomerne krátkom čase. PSO má jednoduchú implementáciu a samotný beh algoritmu závisí na pomerne malom množstve parametrov. Vhodným nastavením vektora rýchlosti môžeme do určitej miery predísť predčasnej konvergencii k lokálnemu optimu a tak nájsť globálny extrém.

3 Ant Colony Optimization

3.1 Úvod

Ant Colony Optimization, ďalej skrátene ACO, je optimalizačná metóda, ktorá sa nechala inšpirovať chovaním mravčích kolónií. Metódu ACO prvýkrát popísal na začiatku 90-tých rokov M. Dorigo s kolegami a uviedol ju ako biologicky motivovanú metaheuristiku na riešenie náročných kombinatorických problémov. Dôkladný prehľad o tejto metóde nájdeme v knihe [1], z ktorej je prevzatá podkapitola 3.2 Experiment Dvojitý most spolu s obrázkami 2 a 3 a podkapitola 3.3 Prechod od živých mravcov k umelým.

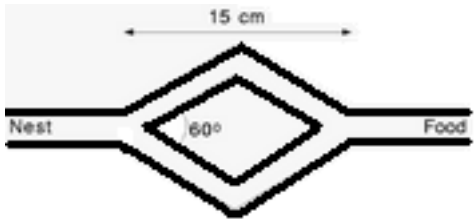
Mravčie kolónie sú systémy, ktoré napriek jednoduchosti jedincov predstavujú vysoko štrukturovanú organizáciu. Ako výsledok tejto organizácie mravčie kolónie vedia vyriešiť také úlohy, ktoré v niektorých prípadoch prevyšujú schopnosti jedinca. Mravce koordinujú svoje aktivity pomocou nepriamej komunikácie sprostredkovanej modifikáciou prostredia, v ktorom sa pohybujú. Napríklad mravec hľadajúci potravu kladie na pôdu chemikáliu, čím zvyšuje pravdepodobnosť, že ďalšie mravce budú nasledovať rovnakú cestu. Pomocou takejto nepriamej komunikácie mravčie kolónie dosahujú samoorganizáciu. Základným prvkom nepriamej komunikácie mravcov je feromónová stopa, ktorá sa používa na vyznačenie cesty napríklad z miesta s potravou do mraveniska.

Algoritmus ACO nepatrí medzi exaktné algoritmy, ktoré môžu garantovať nájdenie optimálneho riešenia a dokázať jeho optimalitu pre všetky konečné inštancie kombinatorických problémov. Na druhej strane v prípade $\mathcal{NP}$-ťažkých problémov exaktné algoritmy počítajú takéto problémy s exponenciálnou zložitosťou v najhoršom prípade. Algoritmus ACO obetoval garanciu nájdenia optimálneho riešenia a môže dosiahnúť suboptimálne riešenie v polynomiálnom čase pomocou rôznych heuristických metód.

V nasledujúcej podkapitole si predstavíme experiment s dvojitým mostom, kde na jednej strane položíme mravce, ktoré budú mať dve možnosti, ako prejsť cez most a dostať sa k potrave, ktorá je na druhej strane mostu. Tieto možnosti budú najprv rovnocenné, t.j. obe cesty k potrave budú rovnako dlhé, a v druhom pokuse bude jedna cesta dvakrát dlhšia ako druhá. V poslednom pokuse sprístupnime najprv len dlhšiu cestu a po čase aj tu kratšiu k potrave. Priníp výberu cesty, po ktorej budú chodiť v záverečnej fáze pokusu takmer všetky mravce, nám pomôže odvodiť algoritmus ACO, ktorý dokáže nájsť najkratšiu cestu.

3.2 Experiment Dvojitý most

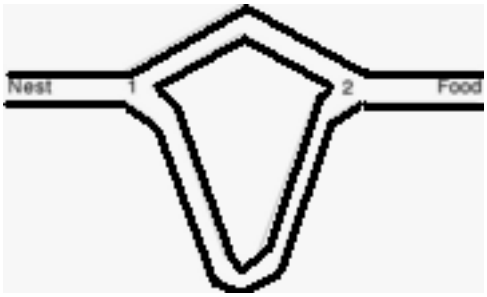
Správanie mravca počas hľadania potravy je založené na nepriamej komunikácii sprostredkovanej pomocou feromónov. Počas cesty z miesta s potravou do mraveniska a naspäť mravce kladú feromóny na zem čím vytvárajú feromónovú stopu. Mravce dokážu cítiť feromóny a majú tendenciu vybrať si cestu označkovanú silnou feromónovou koncentráciou. Predstavíme si experiment, ktorý vytvorili Deneubourg s kolegami, ktorý je popísaný v práci [1]. Majme mravenisko a zdroj potravy, ktoré sú od seba oddelené dvojitým mostom. Najprv si predstavme situáciu, kde obe vetvy mosta sú rovnakej dĺžky, ako na obrázku 2.



Obrázok 2: Príklad dvoch ciest rovnakej dĺžky

Na začiatku sa mravce voľne pohybovali medzi mraveniskom a zdrojom potravy, pričom si vetvy mostu volili náhodne. Z experimentu sa odpozorovalo, že napriek náhodnému výberu vetvy dvojitého mostu, v záverečnej fáze experimentu všetky mravce použili rovnakú vetvu, čo môže byť vysvetlené nasledovne. Pred pustením mravcom na dvojitý most, žiadna z vetví neobsahovala feromóny. Preto mravce nepreferovali žiadnu z vetiev a vyberali si náhodne. Ako dôsledok náhodných fluktuácií po jednej vetve šlo istý čas viac mravcov ako po druhej. Pretože počas chôdze mravce kladú na zem feromóny, tak väčší počet mravcov na vetve spôsobí väčšie množstvo feromónov na tejto vetve a toto väčsie množstvo podoruje výber tejto vetvy, až pokým všetky mravce pôjdu po rovnakej vetve.

V druhom experimente uvažujme dĺžku jednej vetvy dvakrát väčšiu ako je dĺžka prvej vetvy, ako na obrázku 3.



Obrázok 3: Príklad dvoch ciest rôznych dĺžok

V tomto prípade vo väčšine pokusov po určitom čase si všetky mravce vybrali kratšiu vetvu. Na začiatku si opäť mravce vyberajú náhodne, pretože vetvy neobsahujú feromóny. Môžeme očakávať, že v priemere polovica mravcov si vyberie kratšiu vetvu a polovica dlhšiu. Náhodné fluktuácie môžu spôsobiť, že po jednej vetve pôjde viac mravcov ako po druhej. Avšak jedna vetva je kratšia ako druhá, tak mravce, ktoré si vybrali kratšiu cestu, prvé dosiahnu cieľ a skôr sa vrátia do mraveniska. Keď sa mravec musí rozhodnúť, ktorou vetvou pôjde, vyššia koncentrácia na kratšej vetve spôsobí výber práve tejto vetvy. Potom sa feromóny na kratšej vetve naakumulujú rýchlejšie ako na dlhej vetve čo spôsobí, že väčšina mravcov si nakoniec vyberie kratšiu cestu. Určité malé množstvo mravcov si vybralo dlhšiu cestu, čo môžeme chápať ako skúmanie prostredia.

Konečne v treťom experimente uvažujme rovnaký most, t.j. s jednou dvakrát dlhšou vetvou ako druhou a s tým rozdielom, že najprv kratšiu vetvu nesprístupnime. Keď ju po 30 minútach sprístupnime, bude vyberána len veľmi zriedkavo a mravce si budú vyberať aj naďalej dlhšiu vetvu. Toto môže byť vysvetlené vysokou koncentráciou feromónov na dlhšej vetve a pomalým vyparovaním feromónov. Trvanlivosť feromónov je porovnateľná s dĺžkou pokusu, čo znamená, že feromóny sa vyparujú pomaly na to, aby umožnili mravcu zabudnúť na suboptimálnu vetvu.

Podľa opisovaných experimentov Deneubourg s kolegami navrhol jednoduchý pravdepodobnostný model, ktorý opisuje situáciu na dvojitom moste. V tomto modele ψ mravcov za sekundu prejde most konštantnou rýchlosťou v cm/s, pričom počítame mravce idúce v oboch smeroch a kladú na vetvu jednotku feromónu. Je daná dĺžka l_s kratšej vetvy a dĺžka dlhšej vetvy l_l počítané v centimetroch. Mravec vyberajúci kratšiu vetvu prejde trasu za čas $t_s = l_s/v$ sekúnd. Mravec vyberajúci dlhšiu vetvu prejde trasu za čas $r.t_s$, kde $r = l_l/l_s$. Hodnota $p_{ia}(t)$ určuje pravdepodobnosť, že mravec nachádzajúci sa na mieste

$i \in \{mravenisko, zdroj\ potravy\}$ si vyberie v čase t vetvu $a \in \{s, l\}$, kde s označuje kratšiu vetvu a l dlhšiu. Hodnota $p_{ia}(t)$ je závislá na celkovom počte feromónu $\varphi_{ia}(t)$ na vetve a tento počet $\varphi_{ia}(t)$ závisí na počte mravcov, ktoré prešli vetvu do času t. Hodnota pravdepodobnosti výberu kratšej cesty môže byť vyjadrená ako

$$p_{is}(t) = \frac{(t_s + \varphi_{is}(t))^\alpha}{(t_s + \varphi_{is}(t))^\alpha + (t_s + \varphi_{il}(t))^\alpha},$$

kde podľa experimentov od Deneubourga a kolegov sa odvodila hodnota $\alpha = 2$. Analogicky spočítame hodnotu $p_{il}(t)$, pričom $p_{is}(t) + p_{il}(t) = 1$. V tomto modele sa neuvažovalo vyparovanie feromónov a mravce kládli feromóny na ceste z mraveniska k potrave a aj po ceste naspäť. Pozorovanie živých mravcov ukázalo, že mravce, ktoré kládli feromón len počas cesty z miesta s potravou do mraveniska, nie sú schopné nájsť najkratšiu cestu medzi týmito dvomi miestami.

3.3 Prechod od živých mravcov k umelým

Pomocou vzťahov, ktoré sme odpozorovali z experimentov dvojitého mostu môžeme skonštruovať umelého mravca, ktorý sa bude pohybovať po grafe modelujúcom dvojitý most a bude mať schopnosť nájsť najkratšiu cestu na grafe medzi dvomi bodmi, ktoré určujú mravenisko a miesto s potravou. Uvažujme graf, ktorý pozostáva z dvoch vrcholov reprezentujúcich mravenisko a miesto s potravou, ktoré sú spojené kratšou a dlhšou hranou. Ďalej uvažujme, že mravce sa budú pohybovať v diskrétnom čase, t.j. v jednom kroku sa každý mravec posunie k susednému vrcholu. Počas cesty mravce položia jednotku feromónu na hranu, po ktorej prešli. Mravec na pozícii i si s pravdepodobnosťou $p_{is}(t)$ v čase t vyberie kratšiu cestu a s pravdepodobnosťou $p_{il}(t)$ dlhšiu cestu. Tieto pravdepodobnosti sú závislé na množstve feromónov φ_{ia}, ktoré mravec na pozícii i stretne počas cesty na vetve a :

$$p_{is}(t) = \frac{[\varphi_{is}(t)]^\alpha}{[\varphi_{is}(t)]^\alpha + [\varphi_{il}(t)]^\alpha}, \quad p_{il}(t) = \frac{[\varphi_{il}(t)]^\alpha}{[\varphi_{is}(t)]^\alpha + [\varphi_{il}(t)]^\alpha}.$$

Výpočet feromónovej stopy na oboch vetvách budeme vykonávať nasledovne:

$$\varphi_{is}(t) = \varphi_{is}(t-1) + p_{is}(t-1)m_i(t-1) + p_{js}(t-1)m_j(t-1),$$

$$\varphi_{il}(t) = \varphi_{il}(t-1) + p_{il}(t-1)m_i(t-1) + p_{jl}(t-r)m_j(t-r),$$

kde $i, j \in \{mravenisko, zdroj\ potravy\}$ a $i \neq j$. Hodnota $m_i(t)$ označuje počet mravcov na vrchole i v čase t a je daná vzťahom

$$m_i(t) = p_{js}(t-1)m_j(t-1) + p_{jl}(t-r)m_j(t-r),$$

kde $i, j \in \{mravenisko, zdroj\ potravy\}$ a $i \neq j$.

Tento model sa oproti spomínanom pravdepodobnostnom modele líši v tom, že uvažujeme priemerné chovanie celého systému a nie pravdepodobnostné chovanie jednotlivých

mravcov. Ďalej, tento model uvažujeme v diskrétnom čase, pričom predošlý sme uvažovali v spojitom.

Naším cieľom je definovať algoritmus, ktorý dokáže vyriešiť problém najkratšej cesty v grafe. Uvážme graf $G = (V, E)$, kde V je množina vrcholov grafu, $|V| = n$ a E je množina neorientovaných hrán, ktoré sú ohodnotené číslom, ktoré vyjadruje ich dĺžku. Body, medzi ktorými chceme nájsť najkratšiu cestu, nazvime *štart* a *cieľ*.

Ak použijeme umelé mravce odvodené z experimentu dvojitého mostu na riešenie problému najkratšej cesty, tak narazíme na problém - vytváranie cyklov. Cykly sa stanú pre mravcov čím ďalej, tým atraktívnejšie a ostanú v nich uviaznuté. Kladenie feromónov je nevyhnutné pre správny beh algoritmu, preto sa zdá byť odstránenie kladenia feromónov počas cesty z mraveniska k zdroju potravy ako jednoduché riešenie problému so zacyklením. Avšak takýto spôsob odstránenia cyklov, t.j. mravce budú klásť feromóny len na spiatočnej ceste od potravy k mravenisku, zapríčiní, že ani v jednoduchom prípade ako dvojitý most, mravce nebudú schopné nájsť najkratšiu cestu. Riešenie problému eliminácie cyklov bude spomenuté nižsie v tejto kapitole.

Schopnosti mravcov pre vznik cyklov musíme rozšíriť. Priradíme umelému mravcu pamäť, ktorú využijeme pri riešení problému najkratšej cesty v grafe a pridáme mu nasledujúce vlastnosti. Budeme rozlišovať, či mravec ide po ceste z mraveniska k miestu potravu, alebo naspäť, pričom mu dovolíme klásť feromóny len na svojej spiatočnej ceste z miesta potravy k mravenisku.

Využitie pamäti dovoluje mravcu vrátiť sa po rovnakej ceste, po ktorej šiel z mraveniska k miestu s potravou. Pred cestou z miesta s potravou do mraveniska, ktorá je uložená v pamäti, môžeme eliminovať všetky cykly z tejto cesty. Počas cesty do mraveniska mravce budú ukládať na zem feromóny. Mravce si môžu zapamätať aj dĺžku hrán, po ktorých sa pohybovali. Tak môžu ohodnotiť dĺžku cesty, ktorú vygenerovali a zohľadniť toto ohodnotenie pri pridávaní množstva feromónov počas cesty do mraveniska. Ak budú mravce nechávať väčšie množstvo feromónov na kratších cestách, tak rýchlejšie nájdu najkratšiu cestu. Nakoniec pridáme ešte jedno pravidlo - vyparovanie feromónov, ktoré môžeme pozorovať u živých mravcov v prírode. Vyparovanie spôsobí zníženie intenzity feromónov. U umelých mravcov si vyparovanie môžeme predstaviť ako zníženie hodnoty, ktorá označuje počet feromónových jednotiek. Vyparovanie feromónov znižuje vplyv feromónov, ktoré mravec položil v začiatočnej fáze, v ktorej sa generovali riešenia horšej kvality.

Každej hrane (i, j) grafu $G = (V, E)$ priraďme hodnotu τ_{ij}, ktorá určuje množstvo feromónu na príslušnej hrane. Mravec prechádzajúci takto ohodnotenou hranou môže hodnotu prečítať alebo prepísať. Na začiatku prehľadávania nastavíme každej hrane konštantnú hodnotu τ_{ij}, napríklad rovnú jednej. Mravec k nachádzajúci sa na vrchole i využíva hodnoty τ_{ij} pre spočítanie pravdepodobnosti p_{ij}^k vybrania nasledujúceho vrcholu j cesty. Táto pravdepodobnosť sa spočíta ako $p_{ij}^k = \tau_{ij}^\alpha / \sum_{l \in \mathcal{N}_i^k} \tau_{il}^\alpha$, ak $j \in \mathcal{N}_i^k$, inak p_{ij}^k je rovné nule. Vysoká hodnota parametra α zvyšuje vplyv inicializácie a náhodných fluktuácii, čo môže pôsobiť nepriaznivo na beh algoritmu. $\mathcal{N}_i^k$ označuje susedstvo mravca k, keď je v bode i. Susedstvo vrcholu i pozostáva zo všetkých vrcholov, ktoré sú priamo spojené hranou s vrcholom i, okrem vrcholu, z ktorého mravec prišiel do bodu i. Takto sa mravce vyhýbajú vráteniu sa do bezprostredne navštíveného vrcholu pred vrcholom i. Iba v prípade, že $\mathcal{N}_i^k$

je prázdne, vrchol, z ktorého sme sa dostali do vrcholu i bude patriť do $\mathcal{N}_i^k$.

Keď sa mravec dostane k miestu s potravou, vráti sa späť do mraveniska po tej istej ceste, po akej prišiel k potrave. Pred tým, ako sa mravec vydá po spiatočnej ceste do mraveniska, musí eliminovať všetky cykly, ktoré vytvoril na ceste z mraveniska k potrave. Procedúra eliminovania cyklov môže byť daná nasledovne. Majme danú postupnosť vrcholov, ktoré daný mravec navštívil na svojej ceste z mraveniska k miestu s potravou. Postupne pre všetky navštívené vrcholy budeme od konca postupnosti smerom k začiatku postupnosti hľadať vrcholy, ktoré mravec navštívil viac ako jedenkrát. Vezmime si napríklad vrchol na $i-$tej pozícii postupnosti a od konca postupnosti smerom k začiatku budeme hľadať tento vrchol na inej pozícii v postupnosti. Ak ho nájdeme na $j-$tej pozícii, pričom $i < j$, tak môžeme odstrániť cyklus $(i+1, \ldots, j)$. Napríklad ak máme postupnosť navštívených vrcholov $(0, 1, 3, 4, 5, 3, 2, 8, 5, 6, 9)$ a $i = 3$, tak nájdeme $j = 6$ a eliminujeme cyklus $(3, 4, 5, 3)$ a vznikne postupnosť vrcholov $(0, 1, 3, 2, 8, 5, 6, 9)$. Môžeme si všimnúť, že sme neeliminovali najdlhší cyklus $(5, 3, 2, 8, 5)$, ktorý sme porušili eliminovaním kratšieho cyklu. Eliminovaním dlhšieho cyklu by sme dostali kratšiu postupnosť navštívených vrcholov. Najčastejšie je eliminácia cyklov implementovaná tak, že eliminujeme cykly v rovnakom poradí, v akom boli vytvorené.

Počas cesty z miesta s potravou k mravenisku kladie $k-$tý mravec množstvo $\Delta\tau^k$ feromónov na práve navštívenú hranu grafu. Konkrétne, ak je $k-$tý mravec na hrane (i, j), zmení na tejto hrane hodnotu feromónu τ_{ij} nasledovne: $\tau_{ij} := \tau_{ij} + \Delta\tau^k$. V najjednoduchšom prípade môžu byť hodnoty $\Delta\tau^k$ pre všetky mravce konštantné, ale je výhodné túto hodnotu zvýšiť v prípade mravcov, ktoré sa pohybujú po kratších cestách, ako tie ostatné.

Zníženie intenzity feromónov prispieva k prehľadávaniu odlišných ciest a k zabúdaniu nevýhodných - dlhých ciest navštívených v minulosti. Po každom navštívení hrany na spiatočnej ceste z miesta s potravou do mraveniska sa na hodnotu τ_{ij} príslušnú navštívenej hrane aplikuje vyparovanie feromónov podľa vzorca $\tau_{ij} := (1 - \rho)\tau_{ij}$, $(i, j) \in E$, kde $\rho \in (0, 1]$ je parameter. Po aplikácii vyparovania feromónov na všetky hrany, aplikujeme spomínaný vzorec na zvýšenie intenzity feromónov $\tau_{ij} := \tau_{ij} + \Delta\tau^k$.

3.4 Popis algoritmu

Neformálne si môžeme algoritmus ACO predstaviť ako súhru troch procedúr: (1) Konštruovanie riešenia,(2) Výpočet hodnôt, ktoré predstavujú množstvo feromónu,(3) Centrálne riadené akcie, ktoré nemôžu byť vykonané jediným mravcom.

Prvá procedúra slúži na vygenerovanie cesty z mraveniska k miestu s potravou, ktorú si reprezentujeme ako riešenie optimalizačného problému. Generovanie cesty je založené na pravdepodobnostnom výbere pokračovania cesty, pričom velkosť tejto pravdepodobnosti závisí na koncentrácii feromónovej stopy a prípadne na ďalších heuristických informáciach. Pomocou druhej procedúry vypočítame hodnoty τ_{ij}. Najprv necháme vyprchať časť feromónov a potom na prejdené cesty pridáme nové feromóny. Posledná procedúra je určená na vykonávanie akcií, ktoré nemôžu byť vykonané jediným mravcom ako napríklad dovolenie pridania feromónov na nájdenej ceste len tým mravcom, ktoré našli kratšiu cestu ako väčšina ostatných.

Algoritmus ACO je predovšetkým určený pre riešenie optimalizačných problémov, ktoré môžeme redukovať na hľadanie najkratšej cesty v grafe. Teraz popíšeme algoritmus, ktorý je prevzatý z článku [6].

Kombinatorický optimalizačný problém, ktorý sa budeme snažiť vyriešiť pomocou metódy ACO, budeme chápať ako úlohu nájsť také $x \in S$, že $f(x)$ je optimálna hodnota, kde x je *riešenie* optimalizačného problému, S je množina prípustných riešení a f sa nazýva *fitness funkcia*, ktorá určuje kvalitu riešenia $x \in S$.

Definícia 3.1. *Pre daný kombinatorický optimalizačný problém rozumieme konštrukčným grafom orientovaný graf $G = (V, E)$, kde V je množina vrcholov a E je množina hrán, spolu s dekódujúcou funkciou Ω, s nasledujúcimi vlastnosťami:*

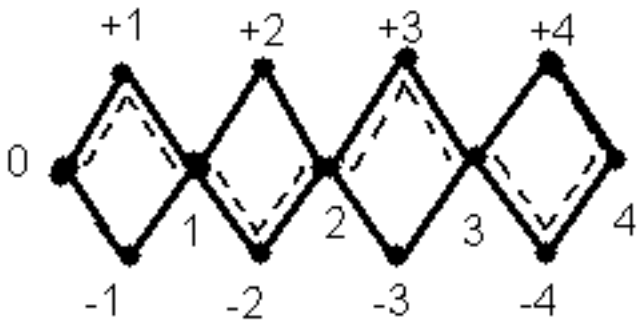
1. *V grafe G máme špeciálny vrchol nazývaný štartovný vrchol.*
2. *Nech W je množina orientovaných ciest $w \in G$ nazývaných prípustné cesty, ktoré splňujú nasledujúce podmienky:*
   (a) *w začína v štartovnom vrchole z G.*
   (b) *w obsahuje každý vrchol z G najviac raz.*
   (c) *w je maximálna cesta vo W v zmysle, že nemôže byť predĺžená pridaním ďalších vrcholov a hrán na dlhšiu prípustnú cestu vo W.*

Funkcia Ω zobrazuje množinu prípustných ciest W do množiny prípustných riešení S.

Môže existovať viac prípustných ciest grafu, ktoré určujú optimálne riešenie x. Cestu v grafe G môžeme chápať ako postupnosť $(v_0, e_1, v_1, e_2, \ldots, e_t, v_t)$, kde $v_0, v_1, \ldots, v_t$ sú navzájom rôzne vrcholy grafu G a pre každé $i = 1, 2, \ldots, t$ je $e_i = \{v_{i-1}, v_i\} \in E$. Definujme si *parciálnu cestu* v grafe G ako cestu u takú, ktorá splňuje podmienky (2a) a (2b). *Pokračovanie* parciálnej cesty $u = (v_0, e_1, v_1, e_2, \ldots, e_t, v_t)$, ktorej posledný vrchol v_t si označme i, si definujme ako hranu $(i, j) \in E$. Hovoríme, že cesta w začína cestou $u = (v_0, e_1, v_1, e_2, \ldots, e_t, v_t)$, ak $w = (u, e_{t+1}, \ldots, v_n)$, kde $n \geq t$. Pokračovanie (i, j) parciálnej cesty u nazvime *prípustné pokračovanie*, ak existuje prípustná cesta $w \in W$ začínajúca parciálnou cestou $u = (v_0, e_1, v_1, e_2, \ldots, e_t, v_t)$ a po ktorej nasleduje hrana (i, j), kde $i = v_t$, t.j. $w = (u, (i, j), j, \ldots)$. Inak toto pokračovanie nazvime *neprípustné*. Neprípustné pokračovanie parciálnej cesty, ktorá končí vrcholom i, si môžeme napríklad predstaviť v probléme obchodného cestujúceho ako hranu (i, j), kde j reprezentuje už navštívené mesto a prípustné pokračovanie je potom hrana (i, j) taká, že je stále možné doplniť túto parciálnu cestu na hamiltonovskú kružnicu obsahujúci všetky zadané vrcholy grafu.

Uvedieme príklad interpretácie dekódujúcej funkcie Ω. Majme nasledujúci konštrukčný graf ako na obrázku 4.

Graf okrem štartovného bodu 0 obsahuje body $1, 2, 3$ a 4, ktoré reprezentujú body množiny $\{1, 2, 3, 4\}$. Pomocou cesty w v tomto konštrukčnom grafe začínajúcej v bode 0



Obrázok 4: Reťazový konštrukčný graf pre n=4

a končiacej v bode 4 môžeme vyjadriť všetky podmnožiny množiny $\{1, 2, 3, 4\}$. Ak cesta obsahuje vrchol $+i$, tak i patrí do uvažovanej podmnožiny $M \subseteq \{1, 2, 3, 4\}$ a ak cesta obsahuje bod $-i$, tak bod i nepatrí do M. Napríklad cesta w označená na obrázku 4 čiarkovane reprezentuje podmnožinu $\{1, 3\}$. Potom dekódujúca funkcia Ω je taká funkcia, ktorá zobrazuje w do M.

Definujme si *feromónové stopy* τ_{ij} ako nezáporné reálne hodnoty priradené hranám konštrukčného grafu. Hodnoty $\eta_{ij}(u)$, použité v algoritme ACO, sa nazývajú *heuristické informačné hodnoty* a tieto hodnoty sa nastavujú v závislosti na danom optimalizačnom probléme. Funkcia g kombinuje feromónové stopy a heuristickú informáciu a je vyjadrená najčastejšie v tvare $g(\tau, \eta) = \tau^\alpha \eta^\beta$, kde parametre α a β sú kladné.

Konečne máme zadefinované všetky potrebné pojmy a môžeme uviesť popis algoritmu ACO, ktorý je na druhej strane.

Procedúra ACO:

1. Inicializujme feromónové stopy τ_{ij} na hranách (i, j) grafu G.
2. Pre iteráciu $m = 1, 2, \ldots$ vykonajme
   (a) Pre mravca $s = 1, 2, \ldots, S$ vykonajme
      i. Nastavme pozíciu mravca i na štartovný vrchol grafu G.
      ii. Parciálnu cestu u nastavme ako prázdny zoznam.
      iii. Pokým prípustné pokračovanie (i, j) cesty u mravca existuje, vykonajme
         A. Vyberme následníka vrchol j s pravdepodobnosťou p_{ij}, kde $p_{ij} = 0$, ak je pokračovanie (i, j) neprípustné, inak

$$p_{ij} = \frac{g(\tau_{ij}, \eta_{ij}(u))}{\sum_{(i,r)} g(\tau_{ir}, \eta_{ir}(u))},$$

            kde suma sa počíta cez všetky prípustné pokračovania (i, r).
         B. Rozšírme cestou u mravca pridaním hrany (i, j) na jej koniec, t.j. $u := (u, (i, j), j)$ a nastavme $i := j$
      iv. Koniec cyklu pre rozširovanie cesty
   (b) Aktualizujme feromónové stopy τ_{ij}
   (c) Ukonči cyklus pre mravca.
3. Ukonči cyklus pre iteráciu m.

3.5 ACO varianty a zhodnotenie ACO algoritmu

Uvedieme dve najznámejšie ACO varianty Ant system a $\mathcal{MAX} - \mathcal{MIN}$ Ant system, ktoré zvyšujú efektivitu pôvodného algoritmu ACO pomocou odlišného spôsobu počítania hodnôt τ_{ij}.

Ant system je charakterizovaný nasledujúcim pravidlom pre výpočet hodnôt feromónových stôp τ_{ij} :

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \frac{\rho}{S} \sum_{s=1}^{S} \Delta\tau_{ij}^{s}.$$

kde parameter $\rho \in [0, 1]$ sa nazýva *faktor vyparovania*. Pri označení w^s mravca s je $\Delta\tau_{ij}^{s}$ rovné $C.f(w^s)$, ak $(i, j) \in w^s$, inak je rovné 0, pričom konštanta C je kladná. V tejto metóde všetky mravce prispievajú k aktualizácii τ_{ij} vo forme feromónového prírastku pre danú iteráciu.

Algoritmus $\mathcal{MAX} - \mathcal{MIN}$ Ant system vylepšuje verziu Ant System v spôsobe aktualizácie feromónu, pri ktorej iba najlepšie mravce môžu pridávať feromón. Hodnotu τ_{ij} spočítame ako

$$\tau_{ij} = [(1 - \rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{best}]_{\tau_{min}}^{\tau_{max}},$$

kde τ_{min} a τ_{max} sú spodná a vrchná hranica hodnoty feromónu. Hodnota $[\tau]_a^b$ je rovná a, ak $\tau < a$, $[\tau]_a^b$ je rovná τ, ak $a \leq \tau \leq b$ a konečne $[\tau]_a^b$ je rovná b, ak $\tau > b$. Hodnota $\Delta\tau_{ij}^{best}$ je definovaná ako $C.f(w^{best})$, ak $(i,j) \in w^{best}$, inak je rovná 0, pričom w^{best} je doteraz najlepší nájdený mravec.

Algoritmus ACO našiel uplatnenie v mnohých kombinatorických optimalizačných úlohách. Podobne ako u metódy PSO aj u metódy ACO je výhodou relatívne malý počet parametrov, ktorých vhodné nastavenie prinesie významné urýchlenie výpočtov. Na druhej strane zase nevýhodou ACO je závislosť na počiatočnom nastavení, ktorá zvyšuje možnosť predčasnej konvergencie v lokálnom optime.

4 Porovnanie BMA s klasickými algoritmami pre problém obchodného cestujúceho

V tejto kapitole sa pokúsime porovnať jeden z biologicky motivovaných algoritmov, konkrétne PSO a jeden z klasických algoritmov, Metódu rezných nadrovín, pri riešení jednej z najznámejších NP-úplných úloh, problému obchodného cestujúceho, ďalej TSP. TSP si môžeme predstaviť ako úlohu obchodného cestujúceho, ktorý má navštíviť daných n miest, pričom každé z nich má navštíviť práve raz a má sa vrátiť do mesta, z ktorého vyšiel a požaduje sa, aby dĺžka celkovej jeho cesty bola čo najkratšia.

4.1 Metóda rezných nadrovín pre TSP

Popis tejto metódy spolu s Tvrdením 4.1 je prevzatý z práce Martina Grötschela, viď [2]. V tejto práci podal teóriu pre riešenie problému TSP a aplikoval ju pre 120 nemeckých miest. Táto podkapitola nemá za úlohu presne popísať spôsob riešenia TSP,ale iba vysvetliť princíp hľadania riešenia a porovnať tento princíp s metódou PSO, špeciálne - využitie náhodnosti. Dôkazy používaných nerovností spolu dôkazom Tvrdenia 4.1 autor neuvádza z dôvodu ich pomerne veľkého rozsahu a čitateľ ich môže nájsť v prácach M. Grötschela [3] a [4].

4.1.1 Terminológia z diskrétnej matematiky

Nech $G = (V, E)$ je neorientovaný graf s množinou vrcholov V a množinou hrán E, ktorý je bez viacnásobných hrán a bez slučiek, kde slučka je hrana, ktorá spája vrchol so samým sebou. Za množinu vrcholov V považujeme množinu $\{1, 2, \ldots, n\}$ a hranu $e \in E$ značíme $\{i, j\}$, kde $i \neq j$. Graf na n vrcholoch je *úplný*, ak $E = \{\{i, j\} : i, j \in V, i \neq j\}$ a značíme ho $K_n = (V, E)$. Postupnosť hrán $C = (\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\})$, kde $k \geq 3$, sa nazýva *kružnica* dĺžky k, ak sú všetky vrcholy $v_i, i = 1, \ldots, k$, navzájom rôzne. Kružnicu dĺžky n nazvime *okružná cesta*, niekedy sa tiež nazýva *hamiltonovská kružnica* a kružnicu dĺžky $k < n$ nazvime *okružná podcesta*. Ak C_1, C_2, ..., C_r sú také kružnice, že každý vrchol je obsiahnutý presne v jednej kružnici C_i, tak zjednotenie týchto kružníc sa nazýva *dokonalé 2-párovanie* alebo jednoducho *2-párovanie*. Potom každá okružná cesta je 2-párovanie. Pre každé $W \subset V$ a $F \subset E$ používame nasledovné značenie:

$$\begin{aligned} V(F) &= \{i \in V : i \text{ je obsiahnuté v hrane } e \in F\} \\ E(W) &= \{\{i, j\} \in E : i \in W, j \in W\}. \end{aligned}$$

Nech $G = (V, E)$ je graf a $\{x_e, e \in E\}$ je množina premenných indexovaná cez množinu hrán E a nech $F \subset E$, $W \subset V$. Potom definujme $x(F) = \sum_{e \in F} x_e$ a $X(W) = x(E(W))$.

Symetrický TSP môžme uviesť takto: Je daný neorientovaný úplný graf $K_n = (V, E)$ a dĺžky hrán $c_e \in \mathbf{R}$ pre všetky $e \in E$. Úlohou je nájsť najkratšiu okružnú cestu v K_n, t.j. takú okružnú cestu T, že $\sum_{e \in T} c_e$ je minimálna.

4.1.2 Popis metódy rezných nadrovín

V nasledujúcej podkapitole sa neformálne povedané budeme snažiť popísať určitú konvexnú množinu, ktorá nám poskytuje riešenie TSP. Avšak úplný popis takejto množiny je veľmi náročný, preto sa budeme snažiť pridávaním rôznych podmienok - nerovníc dostať aspoň jej dobrú aproximáciu, ktorá nám môže poskytnúť riešenie blízke optimálnemu.

Každej hrane $e \in E$ priraďme premennú x_e a každej okružnej ceste $T \subset E$ priraďme incidenčný vektor x^T, t.j. taký vektor, pre ktorý platí, že x_e^T je 1, ak $e \in T$, inak x_e^T je rovné 0. V úplnom grafe je $|E| = \binom{n}{2}$ a označme si túto hodnotu ako m, potom $x^T \in \mathbf{R}^m$. Nech Q_T^n je konvexný obal incidenčných vektorov všetkých okružných ciest, t.j.

$$Q_T^n = conv\{x^T \in \mathbf{R}^m : \text{T je okružná cesta v } K_n\}.$$

Každý vrchol útvaru Q_T^n odpovedá práve jednej okružnej ceste v K_n a naopak. Množinu Q_T^n nevieme alebo je výpočtovo zložité ju presne popísať, preto nájdeme nejakú nadmnožinu $Q \supset Q_T^n$, ktorú už popísať dokážeme v relatívne krátkom čase. Všetky riešenia množiny Q_T^n patria aj do množiny Q, preto TSP môžeme riešiť aj nad množinou Q, ktorá obsahuje spravidla omnoho viac riešení, ako množina Q_T^n a to, či nájdené riešenie bude patriť množine Q_T^n, má náhodný charakter.

Definujme si *m-rozmernú kocku* ako množinu

$$H = \{x \in \mathbf{R}^m : 0 \leq x_e \leq 1 \text{ pre všetky } e \in E\}.$$

Potom z definície Q_T^n vyplýva, že $Q_T^n \subset H$. Okružné cesty a tiež 2-párovanie majú takú vlastnosť, že každý bod v týchto množinách je obsiahnutý v presne dvoch hranách T, preto platí sústava rovníc $Ax = 2e_n$, kde A je incidenčná matica grafu K_n typu n x m, e_n je vektor zložený zo samých jednotiek. Všetky incidenčné vektory okružných ciest a 2-párovania splňujú túto sústavu rovníc. Tak si definujme množinu

$$\overline{Q_{2M}^n} = \{x \in \mathbf{R}^m : Ax = 2e_n, 0 \leq x_e \leq 1 \text{ pre všetky } e \in E\}$$

a potom platí, že $Q_T^n \subset \overline{Q_{2M}^n}$.

Vrcholy útvaru $\overline{Q_{2M}^n}$ sú incidenčné vektory okružných ciest, incidenčné vektory 2-párovania, ktoré nie sú okružné cesty, nazvime ich vrcholy okružných podciest a nakoniec také vrcholy, pre ktoré platí, že $0 < x_e < 1$ pre nejaké $e \in E$, nazvime ich zlomkové vrcholy. Aby sme sa priblížili k útvaru Q_T^n, musíme odstrániť z $\overline{Q_{2M}^n}$ posledné dva typy vrcholov, t.j. vrcholy okružných podciest a zlomkové vrcholy.

Vrcholy okružných podciest odstránime nasledujúcou elimináciou:

$$x(W) \leq |W| - 1 \text{ pre všetky } W \subset V, 2 \leq |W| \leq n - 1. \tag{1}$$

Definujme $Q_S^n = \{x \in \overline{Q_{2M}^n} : x \text{ splňuje } (1)\}$. Potom Q_S^n nemá žiadne celočíselné vrcholy okrem incidenčných vektorov okružných ciest, preto nám ostáva odstrániť už len zlomkové vrcholy.

Na odstránenie zlomkových vrcholov sa používa nasledovná podmienka:

$$\sum_{i=0}^{k} x(W_i) \leq |W_0| + 1/2(k-1), \tag{2}$$

kde množiny vrcholov $W_0, W_1, \ldots, W_k \subset V$ splňujú $|W_0 \cap W_i| = 1, i = 1, \ldots, k$, ďalej $|W_i| = 2, i = 1, \ldots, k, k \geq 1$ a nepárne.

Nech $Q^n_{2M} = \text{conv}\{x^M \in \mathbf{R}^m : M \text{ je 2-párovanie v } K_n\} = \text{conv}\{x \in \overline{Q^n_{2M}} : x \text{ celočíselne }\}$. Potom je dokázane, že $Q^n_{2M} = \{x \in \overline{Q^n_{2M}} : x \text{ splňuje vsetky nerovnosti v (2) }\}$.

Tieto výsledky ukazujú, že práve popísaná eliminácia síce odstráni všetky zlomkové vrcholy v $\overline{Q^n_{2M}}$, ale vytvoria sa nové komplikovanejšie vrcholy, takže pre $Q^n_T \subset Q^n_S \cap Q^n_{2M}$ neplatí rovnosť.

Bolo nájdených niekoľko nerovníc, ktoré platia pre Q^n_T a jedna z najvýznamnejších a najväčších tried týchto nerovníc sa volá trieda hrebeňových nerovníc:

Sú dané $W_0, W_1, \ldots, W_k \subset V$, také, že

$$\begin{aligned} |W_0 - W_i| &\geq 1,\ i = 1, \ldots, k, \\ |W_i - W_0| &\geq 1,\ i = 1, \ldots, k, \\ W_i \cap W_j &= \emptyset, 1 \leq i < j \leq k, \end{aligned}$$

$k \geq 3$ a nepárne. Potom nerovnice platné pre Q^n_T majú tvar:

$$\sum_{i=0}^{k} x(W_i) \leq |W_0| + \sum_{i=1}^{k} (|W_i| - 1) - (k+1)/2 = s(C) \tag{3}$$

Nerovnosť $ax \leq a_0$ platnú pre Q^n_T nazývame *rezná nadrovina* pre Q^n_T, ak

$$\dim(Q^n_T \cap \{x \in \mathbf{R}^m : ax = a_0\}) = \dim Q^n_T - 1.$$

Dve rezné nadroviny sú navzájom ekvivalentné, ak pre $ax \leq a_0$ a $bx \leq b_0$ platí

$$Q^n_T \cap \{x \in \mathbf{R}^m : ax = a_0\} = Q^n_T \cap \{x \in \mathbf{R}^m : bx = b_0\}.$$

Ak K je počet rôznych tried ekvivalentných rezných nadrovín pre Q^n_T a ak vyberieme z každej triedy presne jednu nerovnosť $a^i x \leq a^i_0, i = 1, \ldots, K$, potom platí

$$Q^n_T = \{x \in \mathbf{R}^m : Ax = 2e_n, a^i x \leq a^i_0, i = 1, \ldots, K\}. \tag{4}$$

Táto definícia nie je nadbytočná v zmysle, že ak zoberieme z množiny na pravej strane rovnice (4) akúkoľvek z rovníc $Ax = 2e_n$ alebo akúkoľvek z nerovníc $a^i x \leq a^i_0$, tak novovzniknutý útvar sa už nebude rovnať Q^n_T.

Tvrdenie 4.1. *Nech $n \geq 6$.*

1. *Triviálne nerovnice $x_e \geq 0, x_e \geq 1$ sú rezné nadroviny v Q^n_T pre všetky $e \in E$.*
2. *Nerovnice $x(W) \leq |W| - 1$ na odstránenie vrcholov okružných podciest sú rezné nadroviny v Q^n_T pre všetky $W \subset V, 3 \leq |W| \leq n - 3$.*

3. *Dve rôzne sady nerovníc pre odstránenie vrcholov okružných podciest* $x(W) \leq |W|-1$ *a* $x(W') \leq |W'|-1$ *platné pre* Q_T^n *sú ekvivalentné práve vtedy, keď* $W = V - W'$.

4. *Všetky hrebeňové nerovnice* (3) *sú rezné nadroviny v* Q_T^n.

5. *Dve odlišné sady hrebeňových nerovníc* $\sum_{i=0}^{k} x(W_i) \leq s(C)$, $\sum_{i=0}^{h} x(W'_i) \leq s(C')$ *sú navzájom ekvivalentné pre* Q_T^n *práve vtedy, keď* $k = h, W_0 = V - W'_0, W_i = W'_i, i = 1, \ldots, k$.

6. *Triviálne nerovnice, hrebeňové nerovnice a nerovnice pre odstránenie okružných podciest sú po dvoch neekvivalentné.*

Definujme $Q_C^n = \{x \in \overline{Q_{2M}^n} : x$ splňuje všetky nerovnosti (1) a (3) $\}$, potom zrejme platí:

$$Q_T^n \subset Q_C^n \subset Q_S^n \cap Q_{2M}^n$$

Podľa Tvrdenia 4.1 poznáme všetky rezné nadroviny v Q_C^n. Tvrdenie ďalej hovorí, že všetky rezné nadroviny v Q_C^n sú tiež reznými nadrovinami v Q_T^n. Aj keď nám tvrdenie nepopisuje Q_T^n úplne, Q_C^n je jeho dobrá aproximácia a má celkom jednoduchú kombinatorickú štruktúru. Ďalej nám tvrdenie dáva možnosť skonštruovať algoritmus pre "rezanie nadrovín", ktorý sa vyhýba použitiu nerovníc, ktoré sú si vzájomne ekvivalentné alebo nedefinujú nadroviny.

V prvom rade si zvolíme nejaký útvar Q_1, ktorý obsahuje pomerne málo rezných nadrovín a v ktorom je obsiahnutý Q_T^n a pomocou lineárneho programovania začneme riešit úlohu $\min c.x, x \in Q_1$. Ak máme optimálne riešenie x_1 v Q_1 a je to okružná cesta patriaca do Q_T^n, tak sme vyriešili TSP. V opačnom prípade vyberieme z rezných nadrovín daných v tvrdení sadu nerovníc $Bx \leq b$, ktoré neplatia pre x_1 a pridáme ich do Q_1 a tak odstránime x_1. Potom riešime lineárnym programovaním úlohu $\min c.x, x \in Q_2 = Q_1 \cap \{x : Bx \leq b\}$ a analogicky pokračujeme ďalej.

4.2 Metóda Particle Swarm Optimization pre TSP

Problém obchodného cestujúceho patrí do triedy NP-úplných problémov. V súčasnej dobe sa väčšina matematikov prikláňa k názoru, že polynomiálny algoritmus pre riešenie TSP ani neexistuje. Preto sa skúmajú rôzne heuristiky ako napríklad PSO, ktoré na jednej strane nemusia zaručovať nájdenie optimálneho riešenia, ale na druhej strane sú dostatočne rýchle a dosahujú uspokojivé výsledky. Algoritmus PSO pre problém obchodného cestujúceho nie je potrebné znovu písať, pretože je v princípe rovnaký ako všeobecný popis algoritmu PSO, ktorý je podrobne popísaný kapitole 2.2 s názvom Popis algoritmu. Spomenieme iba základné odlišnosti charakteristické pre zadaný problém oproti všeobecnej verzii PSO ako štruktúra prehľadávaného priestora, štruktúra susedstva, tvar fitness funkcie a výpočet rýchlosti.

Predpokladajme, že máme daných n miest, ktoré reprezentujeme ako vrcholy úplného grafu $K_n = (V, E)$, kde množinu vrcholov uvažujeme ako množinu $\{1, \ldots, n\}$. Potom riešenie TSP je nejaká permutácia na množine $\{1, \ldots, n\}$. Preto si môžeme jedinca reprezento-

vať ako permutáciu a položíme ho do priestoru všetkých permutácii na množine $\{1, \ldots, n\}$. Budeme uvažovať globálnu variantu PSO, t.j. takú, že za susedstvo jedinca sa považuje celá populácia. Pre danú permutáciu $p = (p_1, \ldots, p_n)$ na množine $\{1, \ldots, n\}$ si vzdialenosť medzi jednotlivými mestami p_i a p_j označme ako $d(p_i, p_j)$. Potom hodnotu fitness funkcie f pre danú permutáciu p si definujeme ako

$$f(p) = \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + d(p_n, p_1).$$

Výpočet veľkosti a smeru rýchlosti jedinca umiestneného do diskrétneho priestoru permutácii sa nebude vykonávať tak jednoducho, ako v prípade spojitého euklidovského priestoru, kde len sčítavame príslušné vektory, preto je potrebné metódu PSO modifikovať. Výpočet rýchlosti môžeme vykonať v spojitom euklidovskom priestore a potom vhodnou transformáciou výsledok prevedieme do priestoru permutácii. Uvedieme príklad takejto transformácie spojitého euklidovského priestoru do priestoru permutácii pre problém obchodného cestujúceho, ktorý autor našiel v [7].
Uvažujme priestor permutácii P^n na n-prvkovej množine a spojitý euklidovský priestor $\mathbf{R}^n$ a snažíme sa zostrojiť zobrazenie $f : \mathbf{R}^n \to P^n$ s určitými vlastnosťami. V prvom rade toto zobrazenie musí spĺňať podmienku, že pre ľubovoľný vektor $x \in \mathbf{R}^n$ musí existovať práve jedna permutácia $p \in P^n$, že platí $f(x) = p$. Ďalej zobrazenie f musí nejakým spôsobom využívať vzťah medzi zložkami vektora x, aby sa mohla odvodiť príslušná permutácia p. Pre ľubovoľný bod z $\mathbf{R}^n$ korešpondujúci s vektorom $x = (x_1, \ldots, x_n)$ si permutáciu p takú, že $f(x) = p$, kde $p = (p_1, \ldots, p_n)$, vytvoríme podľa pravidla: $p_k = i$, kde k je počet súradníc vektora $x = (x_1, \ldots, x_n)$, ktoré sú väčšie, alebo rovné ako x_i. Inými slovami vyberieme maximálnu zložku x_i vektora x a jej index sa stane hodnotou prvej zložky p_1 permutácie p. Potom zo zvyšných zložiek vektora x opäť vyberieme maximálnu zložku a jej index sa stane hodnotou p_2, atď. Takouto transformáciou dostaneme z vektoru $x \in \mathbf{R}^n$ príslušnú permutáciu $p \in P^n$.
Zaujímavý princíp výpočtu rýchlosti jedinca v diskrétnom priestore permutácii bez použitia vyššie spomínanej transformácie zo spojitého euklidovského priestoru môžeme nájsť v [8].

4.3 Zhodnotenie algoritmov

Algoritmus PSO patrí medzi heuristické algoritmy, ktoré pri svojích výpočtoch využívajú náhodnosť a nezaručujú nájdenie globálneho optima. Náhodné zložky vystupujú vo vzorci pre výpočet rýchlosti

$$\vec{v_i} = \vec{v_i} + Rand(0, \phi_1) \otimes (p_i - x_i) + Rand(0, \phi_2) \otimes (p_g - x_i).$$

Sú to zložky $Rand(0, \phi_1)$ a $Rand(0, \phi_2)$, ktoré spôsobujú, že sa jedinec náhodne posunie buď smerom k svojej dočasne najlepšej nájdenej pozícii alebo k pozícii svojho najúspešnejšieho suseda. Nevýhodou tohto algoritmu je údržba vektora rýchlosti a náchylnosť k predčasnej konvergencii v lokálnom optime.

Náhodnosť sa do určitej miery vyskytuje i v algoritme rezných nadrovín. Majme nejaký útvar Q, v ktorom je obsiahnutý Q_T^n a pomocou lineárneho programovania hľadáme riešenie $x \in Q$, pričom nás zaujíma len také riešenie, ktoré je obsiahnuté aj v Q_T^n. To, či $x \in Q_T^n$, považujeme za náhodu. Pretože nevieme presne určiť útvar Q_T^n, len blízky útvar Q_C^n, nemôžme zaručiť optimum. Nevýhoda tejto metódy je, že počet nerovníc, ktoré pridávame do podmienky pre nájdenie kvalitnejšieho výsledku, pomerne rýchlo narastá. Pridávaním týchto nerovníc sa blížime k optimálnemu výsledku len veľmi pomaly. V praxi sa pomocou tejto metódy dosahujú výsledky rozdielne od optima len o pár percent a šanca, že nájdeme optimum v relatívne krátkom čase je stále celkom dobrá.

Obe metódy riešenia TSP majú spoločné to, že pri vhodnom nastavení počiatočných podmienok sa významne zníži zložitosť výpočtu. Za vhodné nastavenie počiatočných podmienok u PSO považujeme vybranie nejakým spôsobom úspešnejších jedincov miesto vybrania celkom náhodných jedincov a u metódy rezných nadrovín za vhodné nastavenie počiatočných podmienok považujeme vhodne vybrané platné nerovnosti. V konečnom dôsledku sa oba algoritmy považujú za relatívne efektívne a dosahujú kvalitné výsledky.

5 Možnosti využitia BMA v kryptoanalýze

5.1 Úvod

Biologicky motivované algoritmy PSO a ACO môžeme využiť v mnohých prípadoch, kde je potrebné nájsť optimálne riešenie problému. Nie vždy na prvý pohľad vidíme, že daný problém má optimalizačný charakter. Rovnako dôležitá je správna interpretácia problému a základné nastavenia algoritmov, ako napríklad vhodná štruktúra priestoru, ktorý budeme prehľadávať alebo vhodne definovaná fitness funkcia. Ako príklad použitia biologicky motivovaných algoritmov uvedieme použitie metódy PSO na kryptoanalýzu jednoduchej zámeny.

Jednoduchú zámenu si môžeme zadefinovať nasledovne. Abeceda $\mathcal{A}$ nech označuje množinu písmen medzinárodnej latinskej abecedy $\{A, \ldots, Z\}$, ktorá obsahuje 26 písmen. Predpokladajme, že otvorený text bude pozostávať len z písmen abecedy $\mathcal{A}$. Definujme si permutáciu K na množine $\mathcal{A}$ ako prosté zobrazenie množiny $\mathcal{A}$ na ňu samu. Permutáciu K si môžeme vyjadriť tabuľkou

$$K = \begin{pmatrix} A & B & \ldots & Z \\ r_A & r_B & \ldots & r_Z \end{pmatrix},$$

kde $K(\alpha) = r_\alpha \in \mathcal{A}$ pre každé $\alpha \in \mathcal{A}$. Pri jednoduchej zámene nahradzujeme jednotlivé písmena otvoreného textu príslušnými písmenami abecedy $\mathcal{A}$ a toto nahradenie nám určuje daná permutácia K. Permutáciu K nazývame šifrovací kľúč jednoduchej zámeny. Napríklad pre daný kľúč

$$K = \begin{pmatrix} A & B & C & D & E & F & G & H & I & J & K & L & M & N & O & P & Q & R & S & T & U & V & W & X & Y & Z \\ E & H & X & G & Q & R & A & D & M & O & B & Z & V & T & K & L & C & W & F & I & N & J & P & S & U & Y \end{pmatrix}$$

zašifrujeme otvorený text KRYPTOGRAFIA na text BWULIKAWERME. Dešifrovanie zašifrovaného textu prebieha analogicky ako šifrovanie. Kľúč pre dešifrovanie je inverzná permutácia k permutácii K.

5.2 Popis algoritmu na kryptoanalýzu jednoduchej zámeny pomocou PSO

Základná slabina jednoduchej zámeny spočíva v tom, že vzniknutý zašifrovaný text má rovnakú štruktúru ako pôvodný otvorený text. Bežné kryptoanalytické metódy využívajú frekvenčnú tabuľku unigramov a bigramov a štruktúru slov ako napríklad rozmiestnenie samohlások, či využitie rôznych informácii vyplývajúcich z kontextu. Úspešná kryptoanalýza jednoduchej substitúcie znamená nájdenie šifrovacieho kľúča K, alebo aspoň jeho dostatočne veľkej časti. Keďže šifrovací kľúč K je permutácia, tak nám stačí jednoducho spočítať inverznú permutáciu ku K, pomocou ktorej sa dostaneme k otvorenému textu.

Často nám stačí nájsť 60% otvoreného textu a pre zvyšnú časť môžeme použiť rôzne heuristiky na rozpoznávanie slov, alebo si zvyšok textu jednoducho domyslíme. Preto nám neprekáža, že nám PSO nezaručuje nájdenie globálneho optima.

Riešenie jednoduchej zámeny je permutácia, preto priestor, v ktorom budeme prehľadávať, bude diskrétny priestor permutácii na množine $\mathcal{A}$, označme si ho $\mathcal{P}$. Jedinec bude reprezentovaný ako permutácia $K \in \mathcal{P}$. Susedstvo pre daného jedinca K si definujme ako množinu permutácií $\mathcal{S}_K = \{L \in \mathcal{P} : d(K, L) \leq r\}$, kde $d(K, L)$ vyjadruje vzdialenosť medzi permutáciami K a L a je definovaná ako minimálny počet transpozícií, pomocou ktorých permutáciu K prevedieme na permutáciu L. Parameter r určuje veľkosť susedstva, v ktorom sú obsiahnuté všetky permutácie, pre ktoré stačí najviac r transpozícií, aby sme ich previedli do tvaru permutácie K.

Pri výpočte rýchlosti jedinca budeme potrebovať poznať postupnosť transpozícií, pomocou ktorej permutáciu K prevedieme na permutáciu L. Informácia o tejto postupnosti transpozícií je obsiahnutá v permutácii M, pre ktorú platí $L = M \circ K$, resp. $M = L \circ K^{-1}$, kde operácia $\circ$ značí súčin permutácií. Permutáciu M si vieme rozložiť na súčin transpozícií, čo nám umožňuje nasledujúca veta.

Lemma 5.1. *Každú permutáciu $K \in \mathcal{P}$ je možné rozložiť na súčin transpozícií. Ak je špeciálne $K = (r_A, r_B, \ldots, r_I)$ cyklus, tak $K = (r_A, r_I) \circ (r_A, r_H) \circ \cdots \circ (r_A, r_B)$.*

Rýchlosť jedinca si môžeme predstaviť ako permutáciu $M = t_1 \circ t_2 \circ \ldots t_k$, kde M nech je rozložené na súčin minimálneho počtu transpozícií t_i, pričom veľkosť rýchlosti bude udávať číslo k a za smer rýchlosti budeme rozumieť postupnosť konkrétnych transpozícií t_i. Po aplikácii rýchlosti M na jedinca na pozícii K sa daný jedinec posunie na pozíciu $L = M \circ K$.

Pri definícii fitness funkcie je užitočné zohľadniť, že štruktúra textu sa po zašifrovaní nemení. To znamená, že monogram s najväčšou frekvenciou v otvorenom texte bude prislúchať monogramu s najväčšou frekvenciou v zašifrovanom texte. Rovnako tento princíp môžeme aplikovať v prípade bigramov. Hodnoty frekvencií pre jednotlivé monogramy a bigramy otvoreného textu síce nepoznáme, ale máme k dispozícii tabuľku s očakávanými frekvenciami jednotlivých písmen v danom jazyku.

Nech K je kandidát na správny šifrovací kľúč. Zašifrovaný text dešifrujeme pomocou kľúča K^{-1} a takto vzniknutý text označme $\mathcal{T}$. Pre každý monogram $j_1 \in \mathcal{A}$ nech $C(j_1)$ značí hodnotu jeho frekvencie v texte $\mathcal{T}$. Podobne nech $C(j_1, j_2)$ značí hodnotu frekvencie bigramu $j_1 j_2 \in \mathcal{A}^2$ v v texte $\mathcal{T}$. Ďalej nech $P(j_1)$ je očakávaná hodnota frekvencie monogramu $j_1 \in \mathcal{A}$ a $P(j_1, j_2)$ nech je očakávaná hodnota frekvencie bigramu $j_1 j_2 \in \mathcal{A}^2$. Potom pre daný kľúč K definujeme fitness funkcie ako

$$f(\overrightarrow{K_i}) = \alpha_1 . \sum_{j_1 \in \mathcal{A}} |P(j_1) - C(j_1)| + \alpha_2 . \sum_{j_1, j_2 \in \mathcal{A}} |P(j_1, j_2) - C(j_1, j_2)|,$$

kde parametre α_1 a α_2 určujú váhu významu podobností frekvencií unigramov, resp. bigramov získaných z texte $\mathcal{T}$ s očakávanými frekvenciami. Z definície fitness funkcie je vidieť, že sa budeme snažiť minimalizovať funkčnú hodnotu.

Nech permutácia K_i vyjadruje pozíciu $i-$tého jedinca, V_i vyjadruje rýchlosť $i-$tého jedinca. Permutácia P_i nech vyjadruje pozíciu, na ktorej mal jedinec najnižšiu hodnotu fitness funkcie a túto hodnotu si označme $pbest_i$.
Teraz môžeme popísať algoritmus PSO pre kryptoanalýzu jednoduchej zámeny.

1. Na vstupe načítame zašifrovaný text a tabuľku očakávaných hodnôt frekvencií monogramov a bigramov pre daný jazyk. Náhodne vygenerujme N permutácií K_i na množine $\mathcal{A}$, ktoré vyjadrujú pozície jedincov. Náhodne vygenerujme pre každého jedinca permutáciu V_i, ktorá bude reprezentovať rýchlosť $i-$tého jedinca.

2. Vonkajší cyklus. Ten ukončíme buď v prípade dostatočne malého rozdielu hodnôt fitness funkcie v najhoršej pozícii a v najlepšej pozícii, alebo dostatočne malej hodnoty fitness funkcie pre daného jedinca, alebo v prípade dosiahnutia maximálneho počtu iterácií algoritmu.

3. Vnútorný cyklus. Ten prebieha cez všetkých N jedincov v danej iterácii.

4. Spočítajme hodnoty $C(j_1)$ a $C(j_1, j_2)$ pre daného jedinca a vypočítajme hodnotu fitness funkcie $f(K_i)$.

5. Porovnajme hodnotu $f(K_i)$ s príslušnou hodnotu $pbest_i$. Ak je hodnota $f(K_i)$ menšia ako hodnota $pbest_i$, tak do premennej $pbest_i$ priraďme $f(K_i)$ a do premennej P_i, priraďme súčasnú pozíciu K_i.

6. Nájdime pozíciu jedinca v susedstve s doposiaľ najmenšou hodnotou fitness funkcie z celého spoločenstva a označme jeho pozíciu ako P_g.

7. Zmeňme rýchlosť a pozíciu $i-$tého jedinca pre nasledujúcu iteráciu.

8. Koniec vnútorného cyklu

9. Koniec vonkajšieho cyklu.

Ešte je potrebné popísať spôsob výpočtu rýchlosti a novej pozície jedinca. V diskrétnom priestore budeme používať odlišnú metódu výpočtu, preto tento algoritmus považujme za určitú modifikáciu PSO. Pri pohybe jedinec, ktorý je na pozícii K_i, zohľadňuje svoju doteraz nájdenú najlepšiu pozíciu P_i, a pozíciu svojho najúspešnejšieho suseda, t.j. pozíciu P_g, a navyše zohľadňuje ešte svoju pôvodnú rýchlosť V_i. Do akej miery sa z pozície K_i priblížime k pozícii P_i, nech určuje náhodná zložka $Rand(0, 1)$, ktorá udáva náhodné číslo z otvoreného intervalu $(0, 1)$. Potom do akej miery sa priblížime k pozícii P_g, nech bude určovať hodnota $1-Rand(0, 1)$. Nájdime postupnosť transpozícií, pomocou ktorej permutáciu P_i prevedieme na permutáciu P_g. Takže hľadáme takú permutáciu M_i, že $P_g = M_i \circ P_i$. Potom $M_i = P_g \circ P_i^{-1}$ a rozložíme ju na najmenší počet transpozícií $M_i = t_k \circ t_{k-1} \circ \cdots \circ t_1$, kde k bude značiť dĺžku postupnosti transpozícií. Spočítame si hodnotu $j = [k.(1 - Rand(0, 1))]$, kde $[m]$ značí celú časť reálneho čísla m a nájdeme všetky permutácie X_i také, že $d(X_i, K_i) = j$. Jedna z týchto permutácií X_i bude nová pozícia jedinca. Túto permutáciu X_i vyberáme buď náhodne, alebo pri výbere zohľadníme pôvodnú rýchlosť jedinca V_i.

5.3 Zhodnotenie a možné vylepšenia výsledku algoritmu

Hodnotu j, ktorá určuje, či sa daný jedinec priblíži k najlepšiemu susedovi s pozíciou P_g alebo k svojej dočasne najlepšej pozícii P_i, môžeme spočítať tak, že zoberieme do úvahy hodnoty fitness funkcie $f(P_g)$ a $f(P_i)$. O koľko percent bude hodnota $f(P_g)$ menšia od hodnoty $f(P_i)$, potom o toľko percent budú permutácie X_i bližšie k permutácii P_g ako k permutácii P_i. Náhodnosť použijeme až pri vyberaní konkrétnej permutácie X_i.

Algoritmus som sa pokúsil naprogramovať v programovacom jazyku Java. Za susedstvo pre daného jedinca K som uvažoval množinu permutácií $\mathcal{S}_K = \{L \in \mathcal{P} : d(K, L) \leq 2\}$. V 6. bode algoritmu je potrebné pre každého jedinca z množiny $\mathcal{S}_K$ spočítať hodnoty $C(j_1)$ a $C(j_1, j_2)$, aby sme mohli spočítať hodnotu fitness funkcie. Pre 6. bod algoritmu priebeh programu trval veľmi dlho. Bolo by potrebné znížiť počet susedov pre daného jedinca a nejakým spôsobom vybrať takých, pre ktorých by algoritmus dosahoval lepšie výsledky.

Nakoniec ešte spomeňme niektoré metódy, ktoré nám môžu pomôcť pri kryptoanalýze jednoduchej zámeny. Veľa kľúčov, ktoré nám text nedešifrujú správne, nám môžu poskytnúť optimálnu hodnotu fitness funkcie a na druhej strane kľúč, ktorý správne dešifruje zašifrovaný text môže mať horšiu hodnotu fitness funkcie ako niektoré ostatné kľúče. Tento prípad je spôsobený tým, že hodnoty frekvencií bigramov a monogramov v danom otvorenom texte nekorešpondujú s očakávanými hodnotami frekvencií príslušných bigramov a monogramov. Preto by bolo vhodné pridať k algoritmu heuristiku na rozpoznávanie slov.

V jednoduchšom prípade, keď poznáme dĺžku slov, môžeme použiť nasledujúcu heuristiku. Definujme funkciu

$$Word(P) = \frac{1}{L}. \sum_{M_P^l \in P} RecognizeWord(M_P^l),$$

kde P označuje dešifrovaný text nejakým kľúčom K_i^{-1} a M_P^l značí slovo dĺžky l z textu P. Funkcia $RecognizeWord(M_P^l)$ vráti hodnotu 1, ak slovo M_P^l je v slovníku, inak vráti hodnotu 0. Všetkých dostatočne úspešných kandidátov K_i^{-1} otestujeme touto funkciou $Word$ a eliminujeme nesprávne kľúče tak, že vyberieme taký kľúč, ktorý bude mať hodnotu funkcie $Word$ najväčšiu.

Len zriedkavo vieme zistiť dĺžku slov. V prípade, že dĺžku slov nepoznáme, možné riešenie by mohlo byť v zachovaní čiastočne zmysluplných kusov textu, ktoré by nám pomohli v dešifrovaní zvyšnej časti textu. Tejto téme sa venuje Rabin-Karpov algoritmus.

V poslednom prípade sa k otvorenému textu môžeme priblížiť tak, že vyskúšame ručne všetkých najlepších kandidátov K_i, ktorých sme izolovali pomocou PSO.

Literatúra

[1] Marco Dorigo a Thomas Stützle, (2004): Ant colony optimization. Cambridge: MIT.

[2] Martin Grötschel**,** (1980): On the symmetric travelling salesman problem: solution of a 120-city problem. Mathematical Programming Study 12 , 61-77.

[3] Martin Grötschel**,** (1979): On the symmetric travelling salesman problem I: inequalities. Mathematical Programming Study 16 , 265-280.

[4] Martin Grötschel**,** (1979): On the symmetric travelling salesman problem II: lifting theorems and facets. Mathematical Programming Study 16 , 281-302.

[5] R. Poli, J. Keneddy, T. Blackwell, (2007): Particle swarm aptimization, an overview. Swarm Intell, 33-57.

[6] W.J. Gutjahr, (2007): Mathematical runtime analysis of ACO algorithms, survey on an emerging issue. Swarm Intell, 59-79.

[7] W. Pang et al., (2004): Modified particle swarm optimization based on space transformation for solving traveling salesman problem. Proc. of 2004 International Conference on Machine Learning and Cybernetics, vol. 4, 26-29, pp.2342 – 2346.

[8] X.H. Shi et al., (2007): Particle swarm optimization-based algorithms for TSP and generalized TSP. Information Processing Letters 103, 169-176.