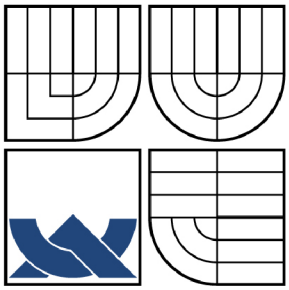
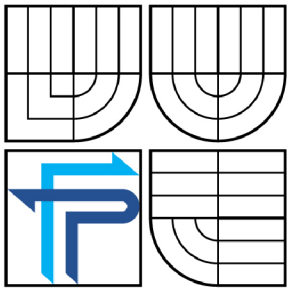


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA PODNIKATELSKÁ

ÚSTAV MANAGEMENTU

FACULTY OF BUSINESS AND MANAGEMENT

INSTITUTE OF MANAGEMENT

APLIKACE OPTIMALIZAČNÍ METODY PSO V PODNIKATELSTVÍ

THE APPLICATION OF PSO IN BUSINESS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. FILIP VESELÝ

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. PETR DOSTÁL, CSc.

SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2009/2010
Ústav managementu

ZADÁNÍ DIPLOMOVÉ PRÁCE

Veselý Filip, Bc.

Řízení a ekonomika podniku (6208T097)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Aplikace optimalizační metody PSO v podnikatelství

v anglickém jazyce:

The Application of PSO in Business

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DAVIS, L. Handbook of Genetic Algorithms, 1. vyd. Int. Thomson Com. Press, USA, 1991, 385s., ISBN 1-850-32825-0.
DOSTÁL, P. Advanced Economic Analyses. 1. vyd. Brno: CERM, s.r.o., 2008, 80s. ISBN 978-80-214-3564-3.
DOSTÁL, P. Pokročilé metody analýz a modelování v podnikatelství a veřejné správě. 1. vyd. Brno: CERM, s.r.o., 2008. 340s. ISBN 978-80-7204-605-8.
MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. Umělá inteligence (4). 1. vyd. Praha: ACADEMIA, 2003, 475s., ISBN 80-200-1044-0.
THE MATHWORKS. MATLAB – Genetic Algorith Toolbox - User's Guide, The MathWorks, Inc., 2008.

Vedoucí diplomové práce: doc. Ing. Petr Dostál, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2009/2010.

L.S.

PhDr. Martina Rašticová, Ph.D. Ředitel ústavu	doc. RNDr. Anna Putnová, Ph.D., MBA

V Brně, dne 21.04.2010

Abstrakt

Tato práce se zabývá dvěma optimalizačními problémy, problémem obchodního cestujícího a shlukovou analýzou. Řešení těchto optimalizačních problémů je aplikováno na potřeby společnosti INVEA-TECH. Práce dále stručně popisuje problematiku optimalizace a některé optimalizační techniky. Podrobněji se zabývá inteligencí roje, přesněji inteligencí částicových hejn. Částí práce je rešerše variant optimalizačních algoritmů na bázi částicových hejn. V druhé části jsou popsány varianty algoritmu PSO řešící problém shlukování a problém obchodního cestujícího a popis jejich implementace v jazyce Matlab.

Klíčová slova

TSP, shlukování, inteligence roje, optimalizace, PSO

Abstract

This work deals with two optimization problems, traveling salesman problem and cluster analysis. Solution of these optimization problems are applied on INVEA-TECH company needs. It shortly describes questions of optimization and some optimization techniques. Closely deals with swarm intelligence, strictly speaking particle swarm intelligence. Part of this work is recherché of variants of particle swarm optimization algorithm. The second part describes PSO algorithms solving clustering problem and traveling salesman problem and their implementation in Matlab language.

Keywords

TSP, clustering, swarm intelligence, optimization, PSO

Bibliografická citace mé práce:

VESELÝ, F. *Aplikace optimalizační metody PSO v podnikatelství.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2010. 66 s. Vedoucí diplomové práce doc. Ing. Petr Dostál, CSc.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 28. května 2010

.....

Filip Veselý

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Petru Dostálovi, CSc. za cenné náměty, připomínky a rady při zpracování této diplomové práce.

Obsah

1 Úvod

Každý ekonomický subjekt, který působí na trhu, musí neustále analyzovat svou činnost a výsledky, kterých dosáhl. Aby podnik neztratil své postavení na trhu, popřípadě jej zlepšil a mohl tak být dále konkurenceschopný, je nutné aby neustále prováděl drobné inovace. Tyto inovace se většinou dějí pomocí optimalizace stávajících firemních procesů, především na straně firemních výdajů a říkáme jim optimalizační úlohy.

Pomocí umělé inteligence se optimalizační úlohy většinou řeší tak, že je na daný problém aplikován určitý postup (algoritmus), jehož sekvenční provádění vede k vyřešení úlohy. Jednotlivé kroky tohoto algoritmu jsou navrženy tak, aby se výpočetní proces neustále přibližoval k nalezení správného řešení.

Rojová inteligence však pracuje na jiném principu, který je inspirován chováním společenství zvířat (především hmyzu). Daný problém je řešen pomocí více jednoduchých prvků, které se navzájem buď přímo nebo nepřímo ovlivňují. Chování těchto prvků je navrženo tak, aby i bez centrálního řízení umožnilo vznik složitějšího chování. Pokud jsou tyto jednotky správně navrženy, jejich vzniklé kolektivní chování převyšuje svými schopnostmi pouhé sjednocení schopností jednotlivých prvků.

Mezi hlavní výhody systémů založených na rojové inteligenci je jejich robustnost a flexibilita. Tedy jsou schopny pokračovat i při selhání některých jednotek a jsou schopny pracovat i za měnících se podmínek. Další předností je také poměrně snadná implementace daná jednoduchou architekturou jednotek.

Kapitola 2 stručně shrnuje cíle této diplomové práce. Kapitola 3 představuje společnost INVEA-TECH, pro kterou byla tato práce zpracována. Stručně popisuje historii a pozadí vzniku firmy. Dále popisuje organizační strukturu firmy, její hospodářské výsledky a analyzuje její současný stav. Kapitola 4 popisuje podstatu optimalizace a optimalizačních problémů a uvádí obecný princip optimalizačních technik. Také jsou zde podrobně popsány dva optimalizační problémy, jejichž řešením se tato práce zabývá. Kapitola 5 představuje samotný algoritmus optimalizace na bázi částicových hejn. Je zde uveden detailní matematický popis algoritmu PSO a jeho binární verze. Také je zde popsáno několik podobných algoritmů. Kapitola 6 popisuje jednotlivé varianty algoritmu PSO.

V 7. kapitole je detailně popsán řešený problém, algoritmy tento problém řešící, jejich implementace a také je zde uveden popis vytvořeného programu. Poslední, 8. kapitola, obsahuje podrobný popis prováděných experimentů a jejich vyhodnocení. Výsledky experimentů jsou přehledně graficky znázorněny.

2 Cíle práce a metody zpracování

Cílem této práce je optimalizovat průběh cest obchodních zástupců a servisních techniků společnosti INVEA-TECH k zákazníkům. Na tomto problému bude ukázána využitelnost optimalizační techniky založené na částicových hejnech (PSO). Flexibilita této metody bude demonstrována na dvou zcela rozdílných optimalizačních problémech. Jsou jimi problém shlukování a problém obchodního cestujícího.

Úkolem tak bude navrhnout a implementovat srozumitelný a přehledný program řešící výše popsané optimalizační problémy. Tento program by měl splňovat požadavek na časovou a uživatelskou nenáročnost. Také musí podávat dostatečně přehledné grafické znázornění výsledného řešení. Program proto bude vytvořen v programovém prostředí a jazyce Matlab.

3 Charakteristika podnikatelského subjektu

Vybraným podnikatelským subjektem, pro který je tato diplomová práce zpracována, je INVEA-TECH a.s. Společnost je zaměřena na vývoj řešení pro vysokorychlostní síťové aplikace.

obrázek 3-1: Logo společnosti INVEA-TECH

Firma sídlí v prostorách Jihomoravského inovačního centra, v Technologickém inkubátoru VUT.

Popis společnosti

Společnost INVEA-TECH je univerzitní spin-off, zaměřený na vývoj nejmodernějších řešení pro vysokorychlostní síťové aplikace. Společnost byla založena v roce 2007 v Brně. Zakladateli jsou fyzické osoby, technologický partner UNIS, Masarykova univerzita a Vysoké učení technické v Brně. Skupina zakladatelů pochází z Masarykovy univerzity, Vysokého učení technického a sdružení CESNET. Sdružení CESNET je vedoucím hráčem ve využívání programovatelného hardwaru pro vysokorychlostní sítě. Jeho výzkumný tým Liberouter, zabývající se vývojem těchto technologií, vznikl v roce 1997 a jeho hlavní výzkumní pracovníci se stali zakladateli firmy INVEA-TECH. Tato společnost má proto více než desetileté zkušenosti v oblasti programování FPGA a mnohaleté zkušenosti s monitorováním vysokorychlostních mezinárodních páteřních linek [8].

Právní forma a předmět podnikání

tabulka 3-1: Výpis z obchodního rejstříku

Datum zápisu	6. června 2007
Obchodní firma	INVEA-TECH a.s.
Sídlo	Brno, Žabovřesky, U Vodárny 2965/2, PSČ 616 00
Identifikační číslo	277 30 450

Právní forma	Akciová společnost
Předmět podnikání	- poskytování software a poradenství v oblasti hardware a software - výzkum a vývoj v oblasti přírodních a technických věd nebo společenských věd - výroba a opravy elektrických strojů a přístrojů a elektronických zařízení pracujících na malém napětí a výroba elektrického vybavení - velkoobchod - specializovaný maloobchod a maloobchod se smíšeným zbožím
Statutární orgán - představenstvo	
předseda představenstva	RNDr. Rostislav Vocilka, CSc.
místopředseda představenstva	Ing. Pavel Čeleda
člen představenstva	Ing. Pavel Valenta
Jednání jménem společnosti	Jménem společnosti jednají vždy dva členové představenstva společně, kde jeden z nich musí být vždy předseda představenstva. Podepisování za společnost se děje tak, že k vytištěné nebo napsaná firmě společnosti připojí svůj podpis společně dva členové představenstva, kde jeden z nich musí být vždy předseda představenstva.
Dozorčí rada	
předseda dozorčí rady	Bc. Jiří Tobola
člen dozorčí rady	Ing. Pavel Jura, CSc.
člen dozorčí rady	RNDr. Ivana Černá, CSc.
Akcie:	2 000 ks kmenové akcie na jméno v listinné podobě ve jmenovité hodnotě 1 000,- Kč
Základní kapitál	2 000 000,- Kč

Právní forma podnikání společnosti INVEA-TECH je akciová společnost. Tato forma je vhodnější pro větší firmy, ale společnost INVEA-TECH očekává rychlý růst a na rozdíl od mnohých společností s ručením omezeným, představuje spolehlivějšího obchodného partnera. Základný vklad společnosti je 2 000 000 Kč. Základní kapitál je rozdělený na 2000 akcií. Každá z nich má hodnotu 1 000 Kč. Převoditelnost akcií je omezena. Akcionáři společnosti mají k akciím, jež mají být předmětem převodu, předkupní právo [14].

Hlavním oborem působení společnosti INVEA-TECH je využití technologie programovatelného hardware (FPGA) v oblasti bezpečnosti a monitorovnání vysokorychlostních sítí se zaměřením na gigabitový a desetigigabitový Ethernet. Společnost nabízí spektrum produktů a služeb zaměřených na analýzu a bezpečnost síťového provozu. Dále společnost nabízí řešení, která pomáhají zákazníkům předcházet obtížím s výpadky sítě či sníženou dostupností a pomalou odezvou kritických aplikací (podnikové systémy, VoIP).

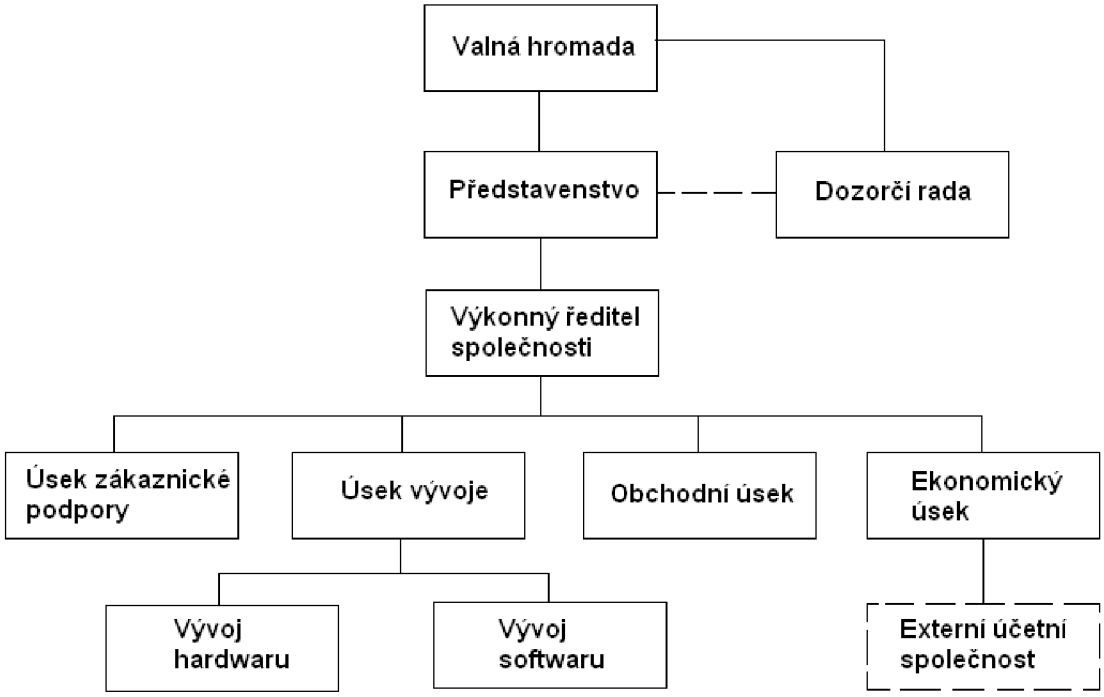
obrázek 3-2: FlowMon sonda - stěžejní produkt společnosti

Mimo tento hlavní obor společnost poskytuje nástroje a technologická zařízení pro vývoj FPGA aplikací. Využívání FPGA technologie umožňuje zákazníkům snadno upravovat jejich produkty a řešení v závislosti na měnících se požadavcích trhu. INVEA-TECH nabízí své bohaté zkušenosti s technologií FPGA a bohatou řadu IP cores, čímž umožňuje svým zákazníkům vyvíjet jejich aplikace s maximální efektivitou.

Organizační struktura

Firma v současnosti zaměstnává jen 10 zaměstnanců, ale během nejbližších let je očekáván personální růst. Hlavním důvodem nízkého počtu pracovníků jsou vysoké požadavky na ně kladené. Firma si totiž zakládá na tom, že každý z jejich zaměstnanců je špičkovým odborníkem.

Vrcholným orgánem společnosti je valná hromada, která se skládá ze všech akcionářů. Majoritním vlastníkem akcií je společnost UNIS. Řídícím orgánem společnosti je představenstvo, jeho předseda je zároveň ředitelem společnosti INVEA-TECH. Dozorčí rada je kontrolním orgánem firmy.



obrázek 3-3: Organizační schéma společnosti

Hospodaření společnosti

Od roku 2007 kdy, byla založena, zaznamenává firma rapidní ekonomický růst. Několikanásobně rostou prodeje a tedy i obrat společnosti. Firma neustále přijímá nové zaměstnance, což lze vysledovat z narůstajících mzdových nákladů. Přehled nejdůležitějších a nejvíce rostoucích ukazatelů je v následující tabulce.

tabulka 3-2: Přehled vývoje hospodaření společnosti (údaje jsou v celých tisících Kč)

Rok	2007	2008	2009
Krátkodobý fin. majetek	1 119	2 443	-
Pohledávky	284	839	-
Závazky	302	884	-
Personální náklady	508	1 256	3 543
Obrat	911	6 115	12 000
Zisk	-824	1 420	2 664

Z rostoucího množství pohledávek a závazků lze usuzovat, že se zvyšuje počet zakázek společnosti. To je potvrzeno také rapidně rostoucím obratem. Ten během dvou let narostl dvanáctinásobně. V roce 2010 dosáhla firma obratu 12 milionů, tedy stejného obratu jako za celý rok 2009, již v měsíci květnu.

Současný stav

Pro zhodnocení současného stavu firmy je velmi vhodným nástrojem SWOT analýza. Je to základní nástroj pro celkovou analýzu vnitřních i vnějších činitelů. S její pomocí je možné komplexně vyhodnotit fungování firmy, nalézt problémy společnosti nebo nové možnosti růstu.

tabulka 3-3: SWOT analýza

Silné stránky	Slabé stránky
Více než desetileté zkušenosti v oblasti. Silné akademické zázemí - výzkum. Dominantní pozice na českém trhu. Mladý flexibilní tým. Špičkový odborníci. Firma nabízí komplexní řešení.	Mladá firma, bez historie a jména. Není ucelená nabídka produktů. Slabý marketing.
Příležitosti	**Hrozby**
Slabá konkurence. Expanze na zahraniční trhy. Poměrně nový stále rostoucí trh. Přijetí vyhlášky č. 485/2005.	Nezvládnutí expanze a růstu firmy. Finanční krize. Vznik nové konkurence. Legislativní změny.

Mezi hlavní silné stránky firmy patří provázanost s akademickou půdou, která pro firmu zajišťuje výzkum nových technologií. Firma z akademické půdy vzešla a proto mají její zaměstnanci velmi bohaté zkušenosti v oboru. Trh, na kterém společnost působí, je velmi mladý, díky čemuž firma rychle dosáhla dominantního postavení na tomto trhu. Konkurence na trhu sice již existuje, ale žádná jiná společnost nenabízí takto rozsáhlá a kompletní řešení jako INVEA-TECH. Za zmínku jistě stojí, že mezi zákazníky firmy patří takové společnosti, jako např. Seznam, Grisoft, Sloane Park, ale také Fakulta podnikatelská na Vysokém učení technickém v Brně.

V současné době firma prochází rapidním růstem. Zvětšují se objemy zakázek a obchodních jednání po celé ČR. Se zvyšujícími se prodeji úzce souvisí také zvětšující se počet servisních zásahů, školení apod. Společnost musí uspokojovat potřeby zákazníků, jejichž počet několikanásobně roste ne každým rokem, ale každým měsícem. Již nejsou výjimkou několikadenní služební cesty po celé ČR. Proto vzniká nutnost tyto cesty k zákazníkům optimalizovat, aby nevznikaly časové ztráty kvůli zbytečnému cestování a také aby byly uspokojeny všechny požadavky v co možná nejkratším čase.

Na rok 2010 je plánována expanze na zahraniční trhy. Právě nezvládnutí této expanze je v současnosti největší hrozbou firmy. Proto se na tento projekt musí zaměřit velký počet pracovníků firmy. Z tohoto důvodu musí být obsluha stávajících zákazníků automatizována a zvládnuta co nejmenším počtem zaměstnanců. To opět vede k nutnosti optimalizace stávajících pracovních cest k zákazníkům.

4 Optimalizace

Úkolem optimalizace je určení hodnot množiny parametrů úlohy tak, aby byly splněny podmínky optimálnosti řešení. Optimalizace je velmi důležitá v mnoha oblastech lidské činnosti, například ve fyzice, chemii, ekonomii či strojírenství. Vědci používají optimalizační techniky při práci s nelineárními křivkami a aproximaci modelů. Využívá se při maximalizaci produkce, minimalizaci nákladů nebo například pro zásobování a přerozdělování zdrojů. Některé tyto problémy lze popsat lineárními modely, jiné však pouze nelineárními modely. Tyto nelineární optimalizační problémy jsou pro řešení velmi složité [2].

Nejjednodušeji se optimalizační problém vyjadřuje jako minimalizace (nebo maximalizace) dané účelové funkce $f(x):A \rightarrow R$. Úkol minimalizace funkce f je ekvivalentní maximalizaci funkce $-f$, proto se nadále v textu nerozlišují.

4.1 Optimalizační techniky

V literatuře lze nalézt velké množství metod, které za určitých podmínek řeší optimalizační problémy. Tyto metody rozlišujeme dle prohledávaného prostoru ($A \subseteq R^n$) a účelové funkce f. Nejjednodušší technikou je lineární programování, které uvažuje lineární funkci f a množinu A omezenou pouze pomocí lineárních rovnic a nerovnic. Obecně však účelová funkce, omezující funkce nebo obě mají v sobě nelinearitu. To zvyšuje důležitost optimalizační techniky zvané nelineární programování, která je výzkumníky pro řešení problémů velmi využívána.

Jelikož je nelineární programování velmi rozsáhlá oblast, odborníci jej dělí na několik případů. Poměrně dobře prozkoumanou oblastí je lineárně omezené programování, kde omezující podmínky jsou určeny lineárně. Pokud je navíc účelová funkce kvadratická, je tento optimalizační problém nazýván kvadratické programování.

Optimalizační úlohy jsou deterministicky definované se všemi známými parametry, ale ve skutečnosti téměř vždy některé parametry neznáme. Proto bylo vytvořeno stochastické programování, které do formulace problému zavádí pravděpodobnostní distribuční funkce jednotlivých proměnných. Nejobecnější stochastická technika se nazývá dynamické programování. Ačkoli je matematicky dokázáno, že metoda dynamického programování najde optimální řešení, má také

několik nevýhod. V mnoha případech je totiž vyřešení algoritmu dynamického programování nemožné. Dokonce i numerické řešení potřebuje ohromující množství výpočetního výkonu, které roste exponenciálně s počtem dimenzí problému. Tyto omezující podmínky vedou k tomu, že algoritmus má malou schopnost předpovědi a nalezené řešení je suboptimální. Navíc stupeň složitosti stoupá ještě víc, pokud se z roviny konečných problémů přesuneme do roviny nekonečných problémů.

Řešením těchto problémů by mohly být na inteligenci založené výpočetní techniky, jako např. genetické algoritmy nebo optimalizace na bázi částicových hejn (PSO). Genetický algoritmus je vyhledávací technika používaná v informatice a strojírenství k nalezení přibližného výsledku optimalizačního problému. Genetické algoritmy představují zvláštní třídu evolučních algoritmů, které využívají techniky inspirované evoluční biologií jako dědičnost, mutaci, přirozený výběr a křížení. Ačkoli dokáží rychle najít dobré řešení i pro velmi složité problémy, mají také několik nedostatků. Pokud není fitness funkce dobře definovaná, může mít genetický algoritmus tendenci konvergovat do lokálního optima namísto globálního. Dále nedokáží dobře pracovat s dynamickými daty a pro některé optimalizační problémy dávají, za stejný výpočetní čas, horší výsledky než jednodušší optimalizační algoritmy [19].

4.2 Optimalizační problémy

Optimalizační problém je každá výpočetní úloha, jejíž zadání obsahuje minimálně tyto atributy. Univerzum všech potenciálních řešení, omezující podmínky, které určují podmnožinu všech přípustných řešení a účelovou funkci, která přiřazuje každému možnému řešení jeho hodnotu. Vyřešením úlohy pak rozumíme nalezení takového přípustného řešení, které dosahuje optimální hodnoty účelové funkce [6].

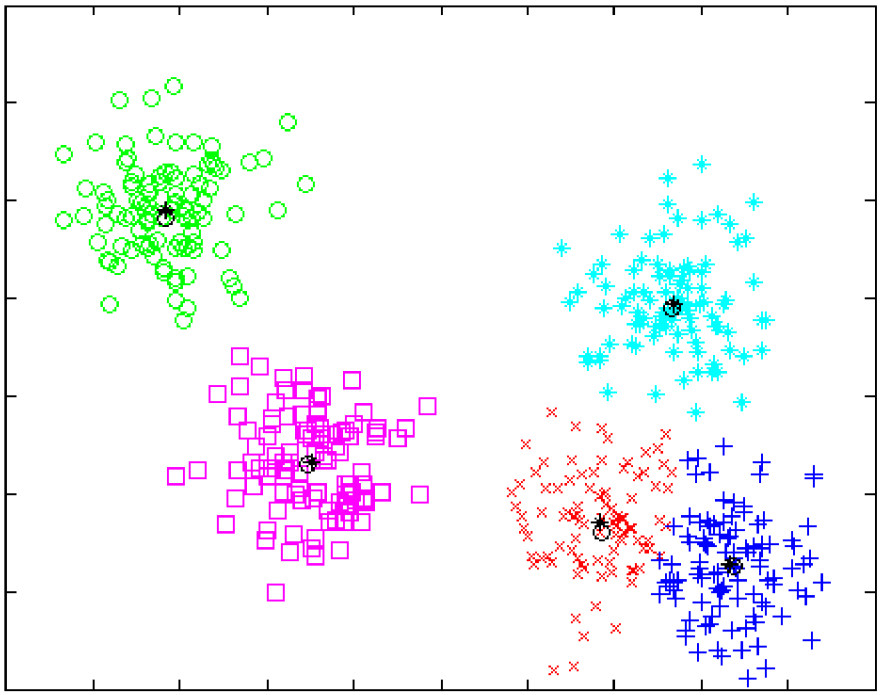
Typických optimalizačních úloh je celá řada. Například optimalizace výroby, problém balení batohu, hledání minimální kostry grafu nebo maximálního toku, SAT problém a mnoho dalších. Blíže popíšeme dvě z nich.

4.2.1 Problém shlukování (clustering)

Problémem shlukování, nebo též shlukovou analýzou rozumíme soubor optimalizačních problémů a metod sloužících k třízení zkoumaných objektů do skupin tak, aby si objekty náležící do stejné skupiny byly podobnější, než objekty ze skupin různých.

Shluková analýza je typický příklad učení bez učitele, protože nepotřebuje žádné předdefinované třídy ani trénovací množinu příkladů [5].

Shlukové analýzy se využívá všude tam, kde je potřeba provést segmentaci dat, rozdělit data do tříd nebo rozpoznat vzory v datech. Největší uplatnění nalézá v biologii, medicíně a marketingu, kde se používá pro rozdělení zákazníku do skupin pro účely marketingového výzkumu. Dále se například používá při zpracovávání obrazu, data miningu, sociálních analýzách apod.



obrázek 4-1: Možný výsledek shlukové analýzy.

Metod shlukové analýzy je celá řada; tyto metody dělíme podle cíle, ke kterému směřují, na hierarchické a nehierarchické. Hierarchické shlukování je sekvence vnořených rozkladů, která na jedné straně začíná triviálním rozkladem, kdy každý objekt dané množiny objektů tvoří jednoprvkový shluk, a na druhé straně končí triviálním rozkladem s jedním shlukem obsahujícím všechny objekty. Formálněji řečeno je hierarchické shlukování systém navzájem různých neprázdných podmnožin, v němž průnikem každých dvou podmnožin je buď jedna z nich nebo prázdná množina a v němž existuje alespoň jedna dvojice podmnožin, jejichž průnikem je jedna z nich. Podle směru postupu při shlukování dělíme metody hierarchického shlukování na aglomerativní a divizivní.

Nehierarchické shlukování je systém navzájem různých neprázdných podmnožin, v němž průnikem každých dvou podmnožin není žádná z nich a tento průnik je prázdný. Jedná se tedy o disjunktní množiny. Proto se metodám nehierarchického shlukování někdy též říká metody založené na rozdělování. Tyto metody lze opět rozčlenit na dva základní typy, na metody založené na centrálním bodu (k-means) a metody založené na reprezentujícím objektu (k-medoids).

Jedním ze základních problémů shlukové analýzy je pojetí vzájemné podobnosti objektů a kvantitativní vyjádření této podobnosti. Tedy stanovení vhodného předpisu d přiřazujícího každé dvojici objektů (i, j) číslo $d(i, j)$, které budeme považovat za míru podobnosti objektů, tak, aby byly splněny požadavky nezápornosti a symetrie. Tj. $d(i, j) \geq 0$ a $d(i, j) = d(j, i)$. Dále by míra podobnosti pro dva stejné objekty měla nabývat maxima z oboru hodnot d. V praxi se však častěji používá nepodobnost objektů neboli vzdálenost. V tom případě by $d(i, j)$ pro $i = j$ mělo nabývat minimální hodnoty, tj. hodnoty nula [10].

Jeden z nejběžnějších způsobů vyjádření podobnostních vztahů mezi objekty jsou metriky vycházející z geometrického modelu dat.

Euklidovská vzdálenost

$$d(i, j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} - x_{j2}\right|^2 + \ldots + \left|x_{in} - x_{jn}\right|^2}, \tag{4-1}$$

Manhattanovská vzdálenost

$$d(i, j) = \left|x_{i1} - x_{j1}\right| + \left|x_{i2} - x_{j2}\right| + \ldots + \left|x_{in} - x_{jn}\right|, \tag{4-2}$$

Minkowského vzdálenost

$$d(i, j) = \left(\left|x_{i1} - x_{j1}\right|^q + \left|x_{i2} - x_{j2}\right|^q + \ldots + \left|x_{in} - x_{jn}\right|^q\right)^{1/q}, \tag{4-3}$$

kde $i = (x_{i1}, x_{i2}, \ldots, x_{in})$ a $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ jsou dva n-dimensionální prvky

Mezi další možné vzdálenostní funkce patří koeficient asociace, korelační koeficient, míra asymetrie, kosinus vektorů, chí-kvadrát apod.

K-means

Tento algoritmus byl poprvé publikován Jamesem MacQueenem v roce 1967. Algoritmus je založen na předpokladu, že n-rozměrné vektory $x = (x_1, x_2, \ldots, x_n)$ tvoří v n-rozměrném prostoru shluky, a že každý shluk i je reprezentován fiktivním centrálním bodem (vektorem „těžiště“ shluku). Metoda dále předpokládá, že počet

shluků k je předem daný, a že do shluků musí být rozděleno všech p vektorů x. Standardní k-means využívá jako podobnostní funkci Euklidovskou vzdálenost.

Algoritmus:

1. Inicializuj k centrálních bodů w (náhodně zvolené vektory z oblasti řešení).
2. Každý vektor x_p přiřaď do shluku C_j, jehož centrální bod w_j má od vektoru x_p nejmenší vzdálenost,

$$d(x_p, w_j) \leq d\left(x_p, w_j\right). \qquad i, j \in \langle 1, k \rangle \tag{4-4}$$

3. Přepočítej všechny centrální body tak, aby byly těžištěm všech vektorů x, které jsou k danému shluku přiřazené,

$$w_j = \frac{\sum_{x_i \in C_j} x_i}{n_j}. \tag{4-5}$$

kde n_j je počet vektorů x ve shluku C_j

4. Vypočítej aktuální chybu shlukování E,

$$E = \sum_{j=1}^{k} \sum_{x_i \in C_j} d\left(x_i, w_j\right). \tag{4-6}$$

5. Pokud chyba E klesla, nebo pokud byl některý z vektorů x přiřazen k jinému shluku C, vrať se na bod 2. Jinak konec [20].

Výhodou tohoto algoritmu je, že je velmi jednoduchý a tvoří velmi kvalitní řešení, pokud data tvoří kompaktní shluky. Navíc má poměrně nízké výpočetní nároky. Mezi nevýhody patří, že je tato metoda citlivá na šum a odlehlé hodnoty, lze ji použít pouze pro metrická data a také že může nalézt více řešení v závislosti na počáteční inicializaci (tzn., že nemusí existovat jediné „správné“ řešení).

K-medoids

Tato metoda principiálně vychází z k-means. Hlavní rozdíl je v tom, že shluk není reprezentován fiktivním centrálním bodem, ale jedním z prvků, který je nejblíže středu daného shluku. Tato metoda se tak daleko lépe vyrovnává se šumem a odlehlými hodnotami. Je však výpočetně složitější, protože je nutné určit, zda má být reprezentant shluku nahrazen jiným prvkem. To znamená, že musíme určit, jak by se změnilo rozdělení prvků do tříd a průměrná vzdálenost objektů od reprezentujícího prvku a na základě tohoto výpočtu provést či neprovést tuto změnu.

Algoritmus:

1. Inicializuj k centrálních bodů w (náhodně zvolené vektory z množiny prvků).
2. Každý vektor x_p přiřaď do shluku C_j, jehož centrální bod w_j má od vektoru x_p nejmenší vzdálenost (4-4).
3. V každém shluku C_j proveď náhradu centrálního bodu w_j každým nereprezentativním vektorem x_p a vypočítej chybu shlukování E (4-6).
4. Vyber konfiguraci centrálních bodů w s nejnižší chybou E.
5. Pokud byl některý ze středů shluků w změněn, vrať se na bod 2. Jinak konec.

Jak již bylo zmíněno výhodou této metody je robustnost vůči šumu a odlehlým hodnotám. Nevýhodou pak větší výpočetní náročnost. Proto se tato metoda hodí spíše pro menší počet rozdělovaných prvků [16].

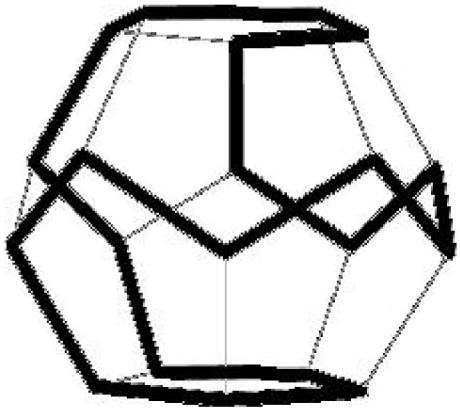
4.2.2 Problém obchodního cestujícího (TSP)

Problém obchodního cestujícího, v angličtině Traveling Salesman Problem (TSP) je obtížný diskrétní optimalizační problém, matematicky vyjadřující a zobecňující úlohu nalezení nejkratší možné cesty procházející všemi zadanými body na mapě. Neformálně lze problém popsat takto. Máme zadáno n měst, kterými musí projet obchodní cestující. Každé město navštíví právě jednou a nakonec se vrací do výchozího města. Hledáme takovou posloupnost měst, která bude reprezentovat nejkratší možnou cestu [7].

Problém nespočívá ani tak ve stanovení postupu nalezení nejkratší cesty – jeden takový postup je totiž skoro samozřejmý: stačí jednoduše prohledat všechny možné uzavřené cesty mezi danými městy a vybrat nejkratší z nich. Problémem však je, že s rostoucím počtem měst (či uzlů grafu) počet možných cest velice rychle narůstá, a tím se doba potřebná k propočtu hrubou silou na počítačích stává zcela neúnosnou už při několika málo desítkách uzlů. Klíčová obtíž je tedy v nalezení časově efektivního algoritmu hledání nejkratších cest.

Historie TSP problému sahá do první poloviny 19. století, kdy byla v díle irského matematika Williama Rowana Hamiltona popsána hra nazvaná The Icosian Game. Tato hra spočívala v pospojování všech vrcholů pravidelného dvanáctistěnu tak, aby byl každý vrchol použit právě jednou (viz obrázek 4-2). Podle toho vznikl pojem hamiltonovská kružnice jako kružnice, která projde právě jednou všemi vrcholy grafu.

W. R. Hamilton splou s britským matematikem Thomasem Penyngton Kirkmanem položili základ oboru matematiky, který se nazývá teorie grafů.



obrázek 4-2: Jedno z možných řešení The Icosian Game.

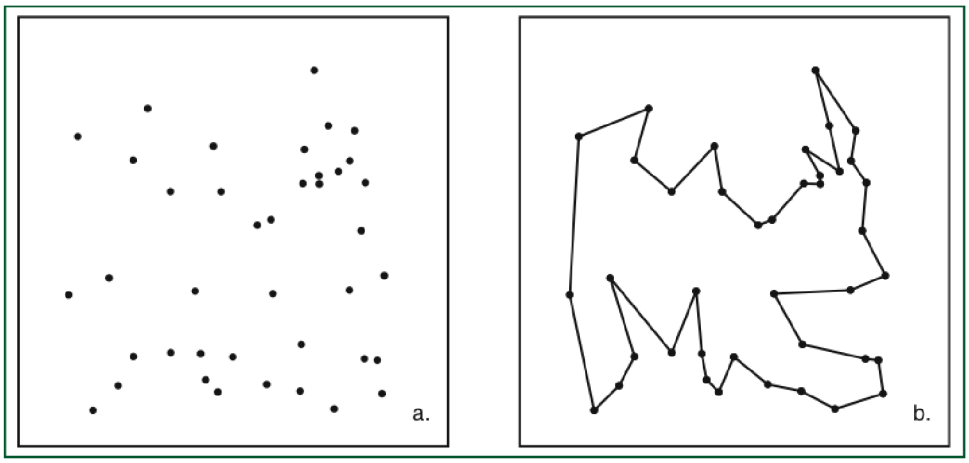
Současný tvar TSP problému byl představem vídeňským matematikem Karlem Mengerem v roce 1930. Od té doby se neustále zvyšuje počet měst, pro něž byl TSP problém vyřešen (od padesátých let především díky schopnostem výpočetní techniky), avšak podstata matematického problému je dosud neobjasněna. V roce 2000 byl Crayovým institutem v USA zařazen mezi sedm nejdiskutovanějších problémů matematiky 21. století. Na jeho řešení, přesněji dokázání či vyvrácení, zdali platí P = NP, byla vypsána odměna 1 mil. USD.

Formálně lze problém popsat pomocí teorie grafů. Uvažujme úplný ohodnocený neorientovaný graf $G = (U, H, c)$, kde $U = \{u_1, u_2, \ldots, u_n\}$ je množina uzlů, $H = \{\{u, v\} \mid u, v \in U \wedge u \neq v\}$ je množina hran (hranu spojující uzly u_i a u_j značíme zkráceně jako h_{ij}) a $c : H \to R$ je cenová funkce, která určuje ohodnocení hran. V tomto grafu budeme hledat hamiltonovskou kružnici vyjádřenou permutací $\pi_{opt} = (u_1, u_2, \ldots, u_n) \in \Pi$, kde $u_i \in U$ pro všechna $i = 1, 2, \ldots, n$ a Π je množina všech hamiltonovských kružnic daného grafu. Pro π_{opt} platí

$$\pi_{opt} = \underset{(u_1,u_2,\ldots,u_n)\in\Pi}{\arg\min} \left(\sum_{i=1}^{n-1} c(h_{ii+1}) + c(h_{n1}) \right), \tag{4-7}$$

Uzly odpovídají městům, hrany představují cesty mezi městy a ohodnocení hran je dáno vzdálenostmi měst. Existuje i asymetrická varianta TSP problému, kde vzdálenosti mezi městy nejsou symetrické. V tomto případě je nutno definovat graf G

jako orientovaný. Další možností je, že neexistují cesty mezi některými městy, v tomto případě je pak graf *G* neúplný [21].



obrázek 4-3: Ukázka zadání a řešení TSP problému.

Obtížnost úlohy je dána velkým stavovým prostorem, protože kompletní graf s *n* uzly obsahuje ½ (*n* -1)! hamiltonovských kružnic. Z teorie složitosti lze vyvodit, že TSP je NP-úplný problém a není tedy znám algoritmus, který by jej dokázal řešit v polynomiálním čase. K řešení menších instancí TSP můžeme využít některý z algoritmů pro hledání přesného řešení. Pro větší počet měst se však používají heuristické algoritmy. Tyto metody nezaručují nalezení skutečně optimálních řešení a jde tedy o kompromis mezi kvalitou řešení a výpočetní náročností. Získaná řešení se však často velmi blíží ideálním řešením při rozumných výpočetních časech. Mezi vhodné heuristické metody pro řešení TSP patří například simulované žíhání, genetické algoritmy nebo algoritmus umělé mravenčí kolonie (ACO).

5 Optimalizace na bázi částicových hejn

PSO je evoluční výpočetní technika vyvinuta Eberhartem a Kennedym v roce 1995, která je inspirovaná sociálním chováním ptačích a rybích hejn. Tato metoda má své kořeny jak v umělé inteligenci či sociální psychologii, tak i v počítačových vědách a inženýrství. PSO využívá populace částic, které prolétají prohledávaným prostorem problému určitou rychlostí. V každém kroku algoritmu je tato rychlost pro každou částici určena individuálně, a to podle nejlepší pozice částice a nejlepší pozice částic v jejím okolí v dosavadním průběhu algoritmu. Nejlepší pozice částic se určí za pomocí uživatelem definované fitness funkce. Pohyb každé částice přirozeně směřuje k optimálnímu řešení nebo k řešení blízkému optimu. V angličtině používaný výraz „swarm intelligence“ (inteligence roje) vychází z nepravidelného pohybu částic v prohledávaném prostoru, připomínající spíše pohyb komárů než ryb nebo ptáků.

PSO je na inteligenci založená výpočetní technika, která není příliš ovlivňována velikostí nebo nelinearitou problému a dokáže konvergovat k optimálnímu řešení i tam, kde většina analytických metod selhává. Navíc má PSO výhody i oproti jiným podobným optimalizačním technikám, např. genetickým algoritmům. Je jednodušší na implementaci a nastavuje se u něj méně parametrů simulace. Každá částice si pamatuje svoji předchozí nejlepší hodnotu a nejlepší hodnotu svých sousedů, proto má efektivnější práci s pamětí než genetický algoritmus. Také účinněji udržuje rozmanitost v populaci, protože částice využívají informace od nejlepší částice ke svému zlepšení, kdežto u genetického algoritmu nejslabší řešení zanikají a pouze ty nejlepší zůstávají do další iterace. To vede k populaci jedinců, kteří jsou velmi podobní tomu nejlepšímu [19].

5.1 Podobné algoritmy

Celulární automaty (CA)

Dá se říci, že ve skutečnosti je PSO jistým rozšířením celulárních automatů. Hejno částic lze reprezentovat jako buňky, jejichž stav se mění na mnoho místech najednou. Společné výpočetní atributy PSO a celulárních automatů jsou následující. Všechny částice (buňky) jsou aktualizovány paralelně. Každá nová hodnota částice (buňky)

závisí pouze na její předchozí hodnotě a na předchozí hodnotě jejích sousedů. Všechny aktualizace se dějí podle stejných pravidel [19].

Optimalizace pomocí mravenčí kolonie (ACO)

Je to jeden z algoritmů založených na rojové inteligenci. Byl vyvinut Marcem Dorigem v roce 1992. Je to pravděpodobnostní technika pro řešení výpočetních problémů, které jdou převést na hledání správných cest v grafu. Je inspirovaná chováním mravenců, kteří hledají cestu od mraveniště k potravě. Ve skutečnosti se na počátku mravenci náhodně procházejí. V momentě kdy najdou jídlo, se s ním vrátí do mraveniště, přičemž za sebou zanechávají feromonovou stopu. Pokud nějaký mravenec na tuto feromonovou stopu narazí, vydá se raději podle ní, než aby chodil náhodně. Pokud na ní nalezne jídlo, tak se po ní neustále vrací a tím tuto feromonovou stopu zesiluje. Časem se však začne feromonová stopa vypařovat, čímž ztrácí svou atraktivitu. Čím je tedy trasa delší, tím déle trvá mravenci přejít ji celou tam i zpět a tím pádem je více času na odpaření feromonu. Naopak kratší cesty jsou procházeny rychleji, a protože je nová vrstva feromonu pokládána rychleji než se stačí vypařit, jeho intenzita zůstává vysoká. Takže když jeden z mravenců nalezne krátkou cestu z mraveniště k potravě, tedy dobré řešení, ostatní mravenci toto řešení následují, což teoreticky vede k tomu, že na konci algoritmu všichni mravenci chodí po stejné cestě. Vypařování feromonů také pomáhá zabránit tomu, aby řešení konvergovalo k lokálnímu optimu. Pokud by k vypařování vůbec nedocházelo, cesty nalezené prvními mravenci by byly pro ostatní příliš atraktivní. Mravenci by procházely pouze tyto cesty, nevytvářely by nové, a tedy by byl prohledávaný prostor řešení omezen.

Idea algoritmu mravenčí kolonie je napodobovat toto chování simulovanými mravenci, kteří se prochází grafem reprezentujícím řešený problém. Tento algoritmus má oproti simulovanému žíhání nebo genetickému algoritmu výhodu v tom, že graf se může dynamicky měnit. Algoritmus mravenčí kolonie totiž může běžet nepřetržitě, protože se dokáže změnám přizpůsobit v reálném čase [7].

Stochastické rozptýlené prohledávání (SDS)

Další z metod rojové inteligence, která je založena na populaci a porovnávání vzorů. Byla představena v roce 1989 Johnem Markem Bishopem. Tato metoda je vhodná pro

optimalizační problémy, které lze rozdělit na několik dílčích nezávislých problémů (účelových funkcí).

Populace je tvořena agenty, kde každý agent prosazuje nějakou hypotézu. Ta je iterativně testována výpočtem náhodně zvolené dílčí účelové funkce, které je parametrizována agentovou hypotézou. Informace o úspěšnosti hypotéz si agenti mezi sebou osobně předávají, čímž jsou rozptýleny napříč celou populací. Časem se agenti dohodnou na jediné hypotéze, která představuje kvalitní řešení daného problému [17].

Umělá včelí kolonie (ABC)

Algoritmus byl představen roku 2005 Dervisem Karabogou. Využívá tří skupin včel, pracujících, dívajících se a zvědů. Možné řešení optimalizovaného problému je reprezentováno jako zdroj potravy, kvalita tohoto řešení (fitness) je dána množstvím nektaru. Náhodná možná řešení problému tvoří populaci a jejich počet je roven počtu pracujících včel. Na začátku algoritmu každá pracující včela letí ke svému zdroji potravy, prozkoumá jeho okolí a vybere to místo, kde je nejvíc nektaru. S touto zapamatovanou pozicí letí zpět do hnízda, kde „zatančí“. Každá dívající se včela si na základě těchto tanců vybere jeden zdroj potravy a letí k němu. Opuštěné zdroje potravy, kam žádná včela neletěla, jsou nahrazeny novými zdroji, které objevili zvědi. Nejlepší zdroj potravy je zaznamenán. To se opakuje, dokud tento nejlepší zdroj nesplňuje požadavky na řešení problému [9].

Algoritmus umělé včelí kolonie kombinuje lokální prohledávací metody (reprezentovány pracujícími včelami) a globální prohledávací metody (zvědi), ve snaze udržet v rovnováze proces vyhledávání a proces využívání [1].

5.2 Základní princip PSO

Každé možné řešení daného problému lze reprezentovat jako částici, která se pohybuje prohledávaným hyperprostorem. Pozice každé částice je dána vektorem x_i a její pohyb je dán rychlostí v_i.

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t). \tag{5-1}$$

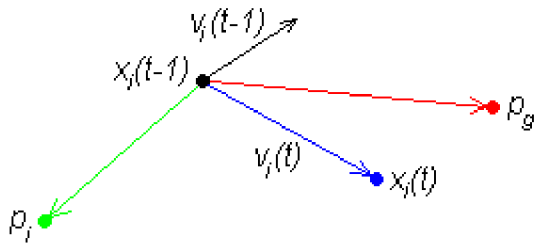
Informace dostupná každé částici je založena na její vlastní zkušenosti a znalosti chování ostatních částic v jejím okolí. Protože důležitost těchto dvou faktorů se může

měnit, je vhodné, aby na každý z nich byla aplikována jiná náhodná váha. Rychlost částice se pak určí následovně

$$\vec{v}_i(t) = \vec{v}_i(t-1) + c_1 \cdot rand_1 \cdot (\vec{p}_i - \vec{x}_i(t-1)) + c_2 \cdot rand_2 \cdot (\vec{p}_g - \vec{x}_i(t-1)), \qquad (5\text{-}2)$$

kde c_1 a c_2 jsou kladná čísla a $rand_1$ a $rand_2$ jsou náhodná čísla v rozmezí 0-1.

Z rovnice aktualizace rychlosti částice (5-2) je patrné, že se skládá ze tří hlavních částí. První část ($\vec{v}_i(t-1)$) se nazývá setrvačnost. Představuje snahu částice pokračovat v původním směru pohybu. Tento parametr může být násoben nějakou váhou. Další částí rovnice je přitažlivost k nejlepší nalezené pozici dané částice p_i, hodnota fitness funkce na této pozici se značí p_{best}. Tato přitažlivost je násobena náhodnou váhou $c_1 \cdot rand_1$ a nazývá se pamětí částice. Poslední třetí částí rovnice je přitažlivost k nejlepší nalezené pozici jakékoliv částice p_g, odpovídající hodnota fitness funkce se značí g_{best}. Tato přitažlivost je opět násobena náhodnou váhou $c_2 \cdot rand_2$ a nazývá se sdílenou informací, nebo též společnou znalostí [19].



obrázek 5-1: Vektorové znázornění aktualizace pozice částice.

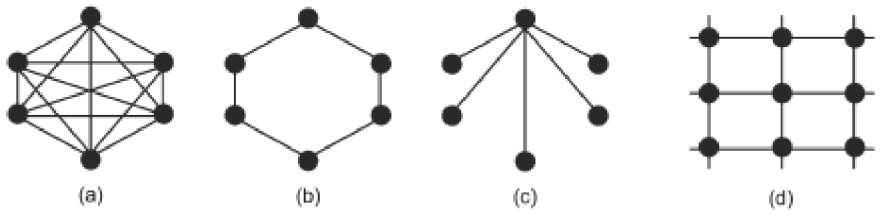
Samotný algoritmus PSO lze shrnout následovně:

1. Inicializace hejna. Každé částici je přiřazena náhodná pozice v prohledávaném hyperprostoru.
2. Pro každou částici je vypočítána hodnota fitness funkce.
3. Porovnání současné hodnoty fitness funkce částice s její p_{best}. Pokud je současná hodnota lepší, je označena za p_{best} a do p_i je uložena současná poloha částice.
4. Nalezení částice s nejlepší fitness funkcí. Tato hodnota je označena za g_{best} a její poloha za p_g.

5. Aktualizace pozic a rychlostí částic dle rovnic (5-1) a (5-2).
6. Opakování kroků 2-5 dokud nejsou splněny podmínky ukončení. Tedy dokud není dosažen maximální počet iterací algoritmu nebo není nalezena dostatečně dobrá hodnota fitness funkce.

Topologie

PSO může fungovat se dvěma základními druhy sousedství. Je to buď globální sousedství, při kterém jsou částice přitahovány k nejlepšímu nalezenému řešení z celého roje. To si lze představit jako plně propojenou síť, kde má každá částice přístup ke všem informacím (viz obrázek 5-2(a)). Druhou možností je lokální sousedství, kde jsou částice přitahovány k nejlepšímu řešení vybíraného pouze z jejich bezprostředních sousedů. U tohoto přístupu existují dvě nejčastější varianty. Jedná se o kruhovou topologii (viz obrázek 5-2(b)), kde je každá částice propojena se dvěma sousedy, nebo o centralizovanou topologii (viz obrázek 5-2(c)). Zde jsou jednotlivé částice od sebe odděleny a veškeré informace jsou shromažďovány v hlavním jedinci.



obrázek 5-2: Topologie rojů.

Předpokládá se, že plně propojená síť konverguje k řešení rychleji, avšak může uváznout v lokálním optimu. Zatímco přístup s omezeným sousedstvím má větší šanci nalézt optimální řešení, ale pomaleji. Dále se předpokládá, že nejlepších výsledků by měl dosahovat roj s von Neumannovskou topologií (viz obrázek 5-2(d)).

Nastavení parametrů

Při implementaci PSO je třeba mít na zřeteli několik předpokladů, aby byla zajištěna konvergence algoritmu a nedošlo k tzv. explozi roje. Mezi tyto předpoklady patří maximální rychlost částic, správné nastavení konstant pro přitažlivost (zrychlení) a nastavení setrvačnosti.

Při každém kroku algoritmu je každé částici vypočítána rychlost pro každý rozměr hyperprostoru řešení. Jelikož je rychlost částice náhodná proměnná, může nabývat jakýchkoliv hodnot a částice se tak může pohybovat chaoticky. Aby k tomu nedocházelo, je vhodné nastavit dolní a horní limit rychlosti částice. Tyto limity je nutno nastavovat empiricky v závislosti na řešeném problému. Pokud by maximální povolená rychlost částice byla příliš vysoká, částice by mohly přeskakovat dobrá řešení. Naopak, pokud by byla příliš nízká, byl by pohyb částic omezen, algoritmus by konvergoval velmi pomalu a nikdy by nemuselo být nalezeno optimální řešení. Výzkum naznačuje, že nejlepším řešením je dynamicky se měnící maximální rychlost (v_{max}).

$$v_{\max} = \left(x_{\max} - x_{\min}\right)/N, \tag{5-3}$$

kde N je uživatelem zvolený počet intervalů v daném rozměru, x_{max} a x_{min} je maximální a minimální hodnota souřadnic dosud nalezených částicemi.

Akcelerační konstanty c_1 a c_2 řídí pohyb částic směrem k nejlepší pozici částice, respektive k nejlepší celkové pozici. Nízké hodnoty těchto konstant omezují pohyb částic, zatímco vysoké hodnoty mohou vést k divergenci algoritmu. Bylo provedeno několik experimentů s jedinou částicí v jednorozměrném prostoru. V tomto případě byla akcelerační konstanta uvažovaná jako jediná, protože nejlepší pozice částice i celého roje je stejná. Tedy $\varphi = c_1 + c_2$. Pro nízké hodnoty φ měla trajektorie charakter sinusoidy. Při vyšších hodnotách se začaly objevovat cyklické trajektorie a při hodnotách vyšších než 4 mířila dráha částice do nekonečna. Po zavedení náhodných vah $rand_1$ a $rand_2$, zamezujících cyklickému opakování dráhy částic, lze obecně říci, že vhodná velikost akcelerační konstanty je právě 4. Tedy $c_1 + c_2 = 4$, neboli $c_1 = c_2 = 2$. Samozřejmě c_1 a c_2 nemusí mít stejnou velikost, opět záleží na řešeném optimalizačním problému.

Zkušenosti ukázaly, že i když jsou maximální rychlost a akcelerační konstanty správně nastaveny, může i tak dojít k explozi roje. V současnosti existují dvě metody snažící se tento problém řešit: omezující faktor a konstanta setrvačnosti.

První z metod je omezující faktor, což je konstanta, kterou se násobí pravá strana rovnice aktualizace rychlosti částice (5-2). Tato konstanta (χ) se používá, pokud je nastaveno $c_1 + c_2 > 4$. Vypočítá se jako:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi}\right|}. \tag{5-4}$$

Výsledná rovnice aktualizace rychlosti se pak zapíše:

$$\vec{v}_i(t) = \chi \cdot \left\{\vec{v}_i(t-1) + c_1 \cdot rand_1 \cdot \left(\vec{p}_i - \vec{x}_i(t-1)\right) + c_2 \cdot rand_2 \cdot \left(\vec{p}_g - \vec{x}_i(t-1)\right)\right\}. \tag{5-5}$$

Obecně omezující faktor zlepšuje konvergenci částic tím, že tlumí oscilaci částic, jakmile se zaměří na nejlepší pozici v oblasti. Nevýhodou je, že částice nikdy nemusí konvergovat, pokud jejich nejlepší pozice p_i je příliš vzdálena od celkové nejlepší pozice hejna p_g.

Druhá metoda pracuje s parametrem, který násobí pouze setrvačnost ($\vec{v}_i(t-1)$) a ne celou pravou stranu jako omezující faktor. Tento parametr se značí φ_{ic}. Rychlost částice se v tomto případě vypočítá jako:

$$\vec{v}_i(t) = \varphi_{ic} \cdot \vec{v}_i(t-1) + c_1 \cdot rand_1 \cdot \left(\vec{p}_i - \vec{x}_i(t-1)\right) + c_2 \cdot rand_2 \cdot \left(\vec{p}_g - \vec{x}_i(t-1)\right). \tag{5-6}$$

Konstanta setrvačnosti může být implementována jako konstantní, nebo se může měnit v průběhu algoritmu. Tento parametr v podstatě kontroluje detailnost prohledávání prostoru optimalizačního problému. Na počátku algoritmu je tato konstanta nastavena na vyšší číslo (obvykle 0,9), takže částice se pohybují rychleji a tedy rychleji konvergují ke globálnímu optimu. Jakmile je nalezena optimální oblast, tato váha se nastaví na nižší (obvykle 0,4). Tím je zvětšena detailnost prohledávání, což napomáhá nalezení opravdového optima. Nevýhodou je, že jakmile je jednou tato váha snížena, roj ztrácí schopnost prohledávat nové oblasti. To může vést k uváznutí v lokálním optimu.

5.3 Binární PSO

V tomto specifickém případě PSO, může každá částice nabývat binárních hodnot TRUE = 1 nebo FALSE = 0. V souladu se sociálním přístupem PSO, lze pravděpodobnost rozhodnutí částice pro 1 nebo 0 zapsat následovně:

$$P(X_{id} = 1) = f\left(X_{id}(t-1), V_{id}(t-1), p_{id}, p_{gd}\right). \tag{5-7}$$

V tomto modelu je pravděpodobnost, že d-tý bit i-té částice nabude hodnoty 1, funkcí předchozího stavu tohoto bitu a mírou sklonu částice k zvolení si 1 nebo 0. Tento sklon je odvozen od individuálního a společného chování částic. Proto pravděpodobnost

$P(X_{id}=1)$ nepřímo závisí i na p_{id} a p_{gd}. V_{id} si lze představit jako práh pravděpodobnostní funkce, a proto by se jeho hodnota měla pohybovat v rozmezí $\langle 0-1 \rangle$. Toho lze dosáhnou pomocí sigmoidální funkce.

$$sig(V_{id}) = \frac{1}{1+\exp(-V_{id})}. \tag{5-8}$$

Pak namísto klasické rovnice aktualizace pozice částice definujeme pravděpodobnostní rovnici:

$$X_{id}(t) = \begin{cases} 1 & \text{jestliže } \rho_{\text{id}} < sig(V_{id}) \\ 0 & \text{jinak} \end{cases}, \tag{5-9}$$

kde ρ_{id} je náhodné číslo z rozsahu $\langle 0-1 \rangle$.

Tato funkce je iterativně vyhodnocena pro všechny dimenze *d* a všechny částice *i*. Rovnice pro aktualizaci rychlosti částic v binárním hejnu je definována následovně:

$$V_{id}(t) = V_{id}(t-1) + c_1 \cdot rand_1 \cdot (p_{id} - X_{id}(t-1)) + c_2 \cdot rand_2 \cdot (p_{gd} - X_{id}(t-1)). \tag{5-10}$$

Tato funkce odpovídá klasické funkci aktualizace rychlosti částic (5-2), bez modifikací jako je omezující faktor nebo konstanta setrvačnosti. Avšak tyto modifikace, stejně jako většina ostatních, jsou aplikovatelné i na binární PSO [19].

6 Varianty algoritmu PSO

6.1 Hybridní PSO

Přirozeným vývojem algoritmu PSO byly pokusy zkombinovat jej s jinými evolučními výpočetními technikami. Mnoho autorů zkouší začlenit do PSO algoritmu výběr, mutaci, křížení a diferenciální evoluci. Hlavním cílem je zvýšení pestrosti populace, buď pomocí samo přizpůsobujících se parametrů, jako je omezující faktor, akcelerační konstanty nebo konstanta setrvačnosti, anebo tím že zabráníme částicím, aby se pohybovaly příliš blízko u sebe. Výsledkem těchto snah bylo vytvoření následujících algoritmů.

Kombinace genetického algoritmu a PSO (GA-PSO)

Tento algoritmus kombinuje výhody rojové inteligence a mechanismů přirozeného výběru tak, že zvyšuje počet dobře hodnocených částic, tím že v každém kroku algoritmu snižuje počet špatně ohodnocený částic. Nejen že lze měnit prohledávané oblasti za pomoci p_{best} a g_{best} parametrů, ale je možné i skákání mezi oblastmi díky mechanismu výběru. To vede ke zvýšení rychlosti konvergence celého algoritmu.

Jedním z možných přístupů jak vylepšit algoritmus PSO, je aplikace reprodukce, která u náhodně zvolených částic mění jak poziční vektor, tak vektor rychlosti. Například takto:

$$\begin{aligned} child_1(x) &= p \cdot parent_1(x) + (1-p) \cdot parent_2(x), \\ child_1(v) &= (parent_1(v) + parent_2(v)) \cdot \frac{|parent_1(v)|}{|parent_1(v) + parent_2(v)|}, \\ child_2(x) &= p \cdot parent_2(x) + (1-p) \cdot parent_1(x), \\ child_1(v) &= (parent_1(v) + parent_2(v)) \cdot \frac{|parent_2(v)|}{|parent_1(v) + parent_2(v)|}, \end{aligned} \tag{6-1}$$

kde p je náhodné číslo z rozsahu ⟨0-1⟩, $parent_{1,2}$(x) reprezentuje poziční vektor náhodně zvolených částic, $parent_{1,2}$(v) představuje odpovídající vektor rychlostí těchto částic a $child_{1,2}$(x) a $child_{1,2}$(v) jsou potomci genetického procesu.

Takto vzniklými částicemi se pak nahradí částice s nízkou hodnotou fitness funkce.

Kombinace evolučního programování a PSO (EPSO)

Evoluční PSO přidává nejen mechanismus výběru, ale také schopnost parametrů se samo přizpůsobit. Jednou z možností je například zavedení turnajového výběru používaného v evolučním programování. Rovnice aktualizace zůstávají stejné jako v klasické verzi PSO. Nejprve se vypočítá hodnota fitness funkce všech částic. Pak je každá částice porovnána s n náhodnými částicemi a započítá se jí bod pokaždé, když je hodnota její fitness funkce větší. Poté je populace seřazena podle tohoto skóre. Současná pozice a rychlost horší poloviny částic je nahrazena hodnotami lepší poloviny, avšak není nahrazena jejich nejlepší známá pozice. Tedy v každém kroku algoritmu je horší polovina částic přesunuta na pozice lepší poloviny s tím, že si zachovají svoji paměť.

Rozdíl mezi tímto přístupem a původním PSO algoritmem je v tom, že je kladen větší důraz na exploatační fázi algoritmu. Díky tomu je globální optimum nalezeno mnohem důsledněji. Rámcově lze algoritmus EPSO popsat takto:

- každá částice je r-krát zkopírována,
- každé částici jsou mutací změněny váhy,
- z každá částice je vytvořeno potomstvo, dle rovnice pohybu částice,
- každý potomek je ohodnocen fitness funkcí,
- nalezení nejlepších částic, které přežijí do další generace, turnajem.

Rovnice pro polohu částic je nezměněna, rovnice pro aktualizaci rychlosti je:

$$\begin{aligned} \vec{v}_i(t) &= w_{i1}^* \cdot \vec{v}_i(t-1) + w_{i2}^* \cdot rand_1 \cdot \left(\vec{p}_i - \vec{x}_i(t-1)\right) + w_{i3}^* \cdot rand_2 \cdot \left(\vec{p}_g^* - \vec{x}_i(t-1)\right), \\ w_{ik}^* &= w_{ik} + \tau \cdot rand. \end{aligned} \tag{6-2}$$

Celková nejlepší nalezená pozice je také změněna mutací:

$$p_g^* = p_g + \tau' \cdot rand, \tag{6-3}$$

kde τ a τ' jsou učící parametry, které mohou být konstantní nebo se dynamicky měnit a w_{ik} jsou strategické váhy (konstanty zrychlení a setrvačnosti).

Kombinace diferenciální evoluce a PSO

Operátor diferenciální evoluce lze pro zlepšení původního PSO použít dvojím způsobem. Lze jej aplikovat na nejlepší nalezenou pozici částice, tak aby neuvízla v lokálním optimu (DEPSO), nebo na nalezení optimálních vah algoritmu (C-PSO).

V předchozích uvedených algoritmech se váhy algoritmu (φ_{ic}, c_1, c_2) určují systémem pokus omyl. Kompositní PSO (C-PSO) je algoritmus, který využívá diferenciální evoluci pro řešení tohoto problému nastavení vah. Lze jej popsat následovně:

1. Inicializace proměnné t (aktuální krok algoritmu) na 1. Nastavení maximálního počtu kroků T. Vygenerování náhodného hejna částic, tj. pozic částic $x(0)$ a jejich rychlostí $v(0)$. Vygenerování parametrů $X = (\varphi_{ic}, c_1, c_2)$, jejichž počet je roven velikosti populace.
2. Vypočítání $v(t)$ a $x(t)$ podle vzorců (5-6) a (5-1) pro každé X. Ohodnocení každé částice fitness funkcí.
3. Aplikace operátorů diferenciální evoluce (mutace, křížení a výběru) na X. Nahrazení X nejlepším vytvořeným X^*. Tento krok se může několikrát opakovat.
4. Opakování kroků 2 a 3, dokud není dosaženo podmínek zastavení algoritmu (dostatečně kvalitní řešení, maximální počet iterací).

6.2 Adaptivní PSO

Dalšími možnými vylepšeními algoritmu je například aplikace fuzzy logiky, Q-learningu nebo použití druhého PSO pro nalezení optimálních vah prvního PSO, řešícího daný optimalizační problém. Dalším problémem, který je nutno řešit, je nalezení optimální velikosti populace hejna a také zvolení vhodné velikosti sousedství.

Tento problém by měl řešit index zlepšení, využívající fitness funkci částice i v čase t, $f(x_i(t))$:

$$\delta(x_i) = \frac{f(x_i(t_0)) - f(x_i(t))}{f(x_i(t_0))}. \qquad (6\text{-}4)$$

Dále je definovaný práh zlepšení, což je limit minimálního přijatelného zlepšení. Existují tři pravidla adaptivní strategie, pro velikost hejna, pro velikost sousedství a pro setrvačnost. I když částice projevuje dostatečné zlepšení, ale je nejhorší ve svém okolí,

je odstraněna. Naopak pokud neprojevuje dostatečné zlepšení, ale je nejlepší ve svém okolí, je vytvořena nová částice. Tím se reguluje velikost hejna. Pokud je částice nejlepší ve svém okolí, ale dostatečně se nezlepšuje, je patrné, že potřebuje více informací, a proto by se mělo zvětšit její sousedství. Pokud se však zlepšuje dostatečně, nemusí kontrolovat tolik ostatních částic, a proto lze velikost jejího sousedství zmenšit. Čím větší je zlepšení částice, tím menší okolí je potřeba prohledat a tím menší by měla být setrvačnost této částice. Opačně pokud je zlepšení částice nedostatečné, tím větší by měla být její rychlost a tedy i setrvačnost.

Z tohoto principu vychází PSO založené na třídách (SBPSO). V této metodě je populace rozdělena na několik tříd podle podobnosti částic. V každé třídě je jedna nejlepší dominantní částice, tzv. semeno. V každém kroku algoritmu jsou nalezena tato semena a označena jako nejlepší částice v okolní skupině částic. To na konci algoritmu vede k nalezení několika lokálních optim, z kterých je vybráno to nejlepší a označeno za hledané globální optimum.

6.3 PSO v komplikovaném prostředí

Víceúčelové PSO (MOPSO)

Víceúčelové optimalizační problémy sestávají z více cílů, kterých je nutno dosáhnout současně. Jedním z možných řešení těchto problémů je shrnutí všech cílů do jediné účelové funkce s váhami, které jsou konstantní nebo dynamicky se měnící. Hlavním problémem tohoto přístupu je, že ne vždy je možné nalézt správně vyváženou funkci. Navíc je někdy nutné nalézt vhodný kompromis mezi jednotlivými cíli, tedy Pareto optimální řešení.

V poslední době bylo představeno několik MOPSO algoritmů založených na Paretově optimu. V těchto algoritmech je hlavním problémem výběr správných nejlepších individuálních hodnot částic (p_{best}) a hodnotu nejlepší částice (g_{best}) tak, aby zaručovaly konvergenci algoritmu do nejslibnější Paretovy oblasti a zároveň byla zachována rozmanitost populace. Tento výběr lze provést dvojím způsobem. Buď náhodně, nebo podle nějakého algoritmu, který nevyužívá náhody, například Paretovy klasifikace, metody sigma nebo podúrovňového stromu.

Jednou z metod řešící víceúčelové optimalizační problémy je PSO s dynamickým sousedstvím (DN-PSO). V tomto algoritmu se cíle rozdělí do dvou množin, cíl skupiny

(F_1) a optimalizační cíl (F_2). Přičemž toto rozdělení je libovolné. V každém kroku si každá částice definuje své okolí tak, že si spočítá vzdálenost k ostatním částicím a zvolí si M nejbližších sousedů. Vzdálenost mezi částicemi je spočítána jako rozdíl v hodnotách fitness funkce první skupiny účelových funkcí. Poté je v takovémto sousedství nalezena částice s nejlepší hodnotou fitness funkce druhé skupiny účelových funkcí.

Další metodou je vektorově ohodnocené PSO (VEPSO), které je založeno na principu vektorově ohodnoceného genetického algoritmu (VEGA). V algoritmu VEPSO je pro prohledávání prostoru řešení použito dvou a více hejn. Každé hejno je ohodnocováno dle jedné účelové funkce, přičemž si mezi sebou vyměňují informace. Díky informacím získaným z jiných hejn se upravuje trajektorie částic. Ty potom směřují k bodu Paretova optima. Rovnice aktualizace rychlosti pro M-účelovou funkci pak může být definována takto:

$$v_i^{[j]}(t) = \chi^{[j]} \cdot \begin{Bmatrix} \varphi_{ic}^{[j]} \cdot v_i^{[j]}(t-1) + c_1 \cdot rand_1 \cdot \left(p_i^{[j]} - x_i(t-1)\right) \\ + c_2 \cdot rand_2 \cdot \left(p_g^{[j]} - x_i(t-1)\right) \end{Bmatrix}, \qquad (6\text{-}5)$$

kde index *j* představuje číslo hejna, index *i* odpovídá číslu částice v hejně.

Tento algoritmus je možno snadno implementovat na paralelních počítačích, v tom případě se pak algoritmus nazývá paralelní VEPSO.

Práce s omezeními v PSO

Skutečné optimalizační problémy často obsahují různá omezení, která vymezují prohledávaný prostor řešení na nějakou vhodnou malou oblast. Způsoby jak nakládat s těmito omezeními jsou dva. Jednou z možností je zahrnout tato omezení do fitness funkce. Druhou možností je řešit omezení a fitness odděleně.

Výhodou druhého přístupu je, že nezavádí do PSO algoritmu další proměnné a navíc není počet ani formát těchto omezení nijak omezen. Základní rovnice pro aktualizaci pozice a rychlosti zůstávají nezměněny. Poté co je částici vypočítána nová pozice, ověří se, zda tato pozice patří do přípustného prostoru řešení. Pokud ne, tak je částice buď znovu inicializována, nebo přesunuta na její nejlepší známou pozici p_i, nebo se počítá dál i s tímto nepřípustným řešením, ale neaktualizuje se hodnota p_{best}.

Nevýhodou této metody je problém s řešením optimalizačních úloh, které mají velmi malou oblast přípustných řešení.

PSO v dynamických úlohách

Klasický optimalizační algoritmus založený na bázi částicových hejn se ukázal při řešení statických optimalizačních úloh jako velmi efektivní. Avšak tato metoda už nemusí být tolik efektivní pro řešení dynamických systémů, kde se optimální hodnota rychle mění. Proto byly vytvořeny přizpůsobivé modifikace PSO algoritmu. Tyto modifikace pracují jak s náhodným přenastavením částic, tak s dynamicky se měnícími parametry samotného PSO algoritmu.

V literatuře se vyskytují dvě metody pro rozpoznání změny prostředí, měnící se g_{best} hodnota a stálá g_{best} hodnota. První metoda v každém kroku algoritmu vypočítá hodnotu fitness funkce pro p_g. Pokud p_g odpovídá stále stejné částici, ale hodnota fitness funkce se liší od g_{best}, pak lze předpokládat, že se systém dynamicky změnil. Protože však tento předpoklad nemusí nutně platit pro všechny dynamické systémy, existuje druhá metoda, která sleduje pozice dvou nejlepších částic hejna. Pokud se totiž jejich pozice za určitý počet iterací nezmění, algoritmus předpokládá, že nalezl optimum. Existuje řada způsobů, jak se s těmito dynamickými změnami systému vypořádat. Například náhodně přenastavit určitý počet částic (10%, 50%, 100% populace), reinicializovat některé částice, náhodně měnit p_g a g_{best}, nebo kombinace předchozích způsobů.

Jiným algoritmem pro dynamické systémy je PSO s malou velikostí populace (SPPSO). Tento algoritmus využívá populace pěti a méně částic, které se každých N kroků znovu vygenerují. Všechny částice, kromě nejlepší, jsou nahrazeny. Navíc jsou částicím předány hodnoty p_i a p_{best}, aby byla zachována paměť algoritmu.

6.4 Další varianty PSO

Gaussovo PSO (GPSO)

Klasický PSO algoritmus provádí vyhledávání uprostřed mezi globální a lokální nejlepší pozicí. Způsob jakým je toto prohledávání prováděno, stejně jako konvergence hejna do optimální oblasti, závisí na tom, jak jsou nastaveny parametry algoritmu. Pro odstranění této slabiny byla pro řízení pohybu částic do PSO implementována Gaussova

funkce. V této variantě PSO již není potřebná konstanta setrvačnosti. Akcelerační konstanty jsou nahrazeny náhodným číslem z Gaussova rozložení.

Rovnice aktualizace rychlosti je pak definována jako:

$$\begin{aligned} |v(t)| &= GRand \cdot (1 - C_1) \cdot |p_i(t-1) - p_g(t-1)| \quad when \quad rand > C_1, \\ |v(t)| &= GRand \cdot (C_2) \cdot |p_i(t-1) - p_g(t-1)| \quad when \quad rand \le C_1, \\ v(t) &= |v(t)| \cdot Rand(\theta), \end{aligned} \tag{6-6}$$

kde $|p_i(t-1) - p_g(t-1)|$ je vzdálenost mezi globální a lokální nejlepší pozicí, pokud jsou oba body shodné, je nastavena na 1, C_1 je konstanta v rozmezí ⟨0-1⟩. Představuje důvěru v globální nejlepší pozici. C_2 je opět konstanta v rozmezí ⟨0-1⟩. Stanovuje bod mezi $p_g(t)$ a $p_i(t)$, který je jejich směrodatnou odchylkou, *GRand(y)* je náhodné číslo z Gaussova rozložení se směrodatnou odchylkou *y*, *rand* je náhodné číslo mezi nulou a jedničkou z rovnoměrného rozložení a *Rand(θ)* je náhodný vektor s velikostí jedna, úhel je mezi nulou a 2π.

Díky této modifikaci je převážně prohledávána oblast mezi lokální a globální nejlepší pozicí. Čím jsou si tyto pozice blíže, tím menší je směrodatná odchylka, a tím se prohledávané oblasti přibližují.

Rozptýlené PSO (DPSO)

DPSO pro zlepšení algoritmu PSO zavádí do modelu zápornou entropii, což vytváří rozptýlené struktury, které zabraňují předčasné stagnaci algoritmu. Negativní entropie dává rychlosti částice dodatečnou náhodnost takto:

$$If(rand < c_v) \quad then \quad v_i = rand \cdot V_{\max}, \tag{6-7}$$

kde c_v a *rand* jsou náhodná čísla v rozmezí ⟨0-1⟩.

Stejným způsobem je přidána náhodnost pozice částice

$$If(rand < c_l) \quad then \quad x_i = Rand(l, u), \tag{6-8}$$

kde *Rand(l,u)* je náhodné číslo v rozmezí daném dolním limitem *l* a horním limitem *u*.

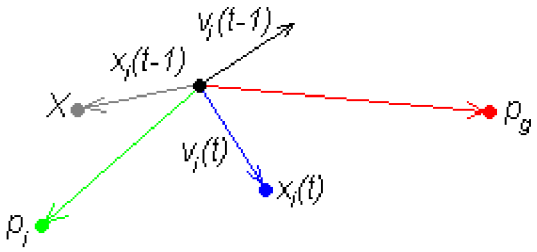
Náhodnost dodaná negativní entropií zabraňuje algoritmu dospět do stavu rovnováhy. Samoorganizace rozptýlených struktur společně s vrozenou nelineární interakcí částic v hejně vede k trvalému kolísání řešení.

PSO s pasivním shromažďováním (PSOPC)

Pasivní shromažďování je jedna z možností jak zabránit uváznutí algoritmu PSO v lokálním optimu a navíc zlepšuje přesnost a rychlost jeho konvergence. Zahrnutím pasivního shromažďování do algoritmu PSO se změní rovnice aktualizace rychlosti částice na:

$$v_i(t) = \varphi_{ic} \cdot v_i(t-1) + c_1 \cdot rand_1 \cdot (p_i - x_i(t-1)) + c_2 \cdot rand_2 \cdot (p_g - x_i(t-1)) + c_3 \cdot rand_3 \cdot (X - x_i(t-1)), \quad (6\text{-}9)$$

kde c_3 je koeficient pasivního shromažďování a X je náhodně zvolená částice z hejna.



obrázek 6-1: Aktualizace pozice částice u algoritmu PSOPC.

Bohužel zatím není zjištěno nakolik tato metoda zlepšuje chování algoritmu, ani jaká je vhodná velikost koeficientu pasivního shromažďování.

Rozpínavé PSO (SPSO)

Schopnost konvergence algoritmu za přítomnosti lokálních optim je jedním z hlavních problémů optimalizačních technik. Může se stát, že řešení uvázne v lokálním optimu, u kterého započalo prohledávání prostoru řešení. Proto bylo vyvinuto rozpínavé PSO, které je zaměřeno na nalezení opravdového globálního optima.

Tato metoda přidává do konceptu PSO techniky ohybu, rozpínání a odpuzování. První dvě techniky provádí transformaci účelové funkce tím, že do ní zahrnují již nalezené optima. Poslední technika zaručuje, že se částice nebudou pohybovat směrem

k již nalezeným optimům. Díky tomu se algoritmus dokáže vyhnout již nalezeným řešením a tedy má větší šanci nalézt globální optimum účelové funkce.

$$G(x) = f(x) + \gamma_1 \|x - \bar{x}\| \cdot (\text{sgn}(f(x) - f(\bar{x})) + 1), \tag{6-10}$$

$$H(x) = G(x) + \gamma_2 \cdot \frac{\text{sgn}(f(x) - f(\bar{x})) + 1}{\tanh(\mu(G(x) - G(\bar{x})))}, \tag{6-11}$$

kde γ_1, γ_2, a μ jsou náhodně zvolené kladné konstanty.

Uvedené rovnice provádí dvoufázovou transformaci fitness funkce. První transformace (6-10) mění fitness funkci *f(x)* na funkci *G(x)* tím, že eliminuje všechna lokální minima, která jsou větší než $f(\bar{x})$, kde $\bar{x}$ představuje právě nalezené lokální minimum. Druhá transformace (6-11) roztahuje sousedství $\bar{x}$ a přiřazuje větší funkční hodnoty bodům v tomto rozšiřujícím se sousedství.

Spolupracující PSO (CPSO)

Tato metoda využívá spolupráci více hejn částic k významnému zlepšení základního PSO algoritmu. Každé z používaných hejn optimalizuje jinou část vektoru řešení. Stejně jako u kooperativního koevolučního genetického algoritmu (CCGA), je prohledávaný prostor jasně rozčleněn rozdělením vektoru řešení na několik menších vektorů. Na tomto principu pracují dva algoritmy, CPSO-S a CPSO-H.

U CPSO-S je hejno n-rozměrných vektorů rozděleno na n hejn jedno-rozměrných vektorů. Každé hejno se pak snaží optimalizovat jednu část vektoru řešení. Pro ohodnocení jednotlivých částic se používá mechanismus důvěrného předání. Výhodou CPSO-S algoritmu je, že současně je modifikována pouze jedna část. Tudíž je možno vytvořit mnoho kombinací pomocí různých částic z různých hejn. Tím je zajištěno podrobné prohledávání a přitom i zvýšení rozmanitosti populace.

PSO algoritmus má větší šanci uniknout z lokálního optima, zatímco CPSO-S má rychlejší konvergenci. Algoritmus CPSO-H kombinuje tyto dva přístupy tak, že nejprve provede jeden krok CPSO-S algoritmu a poté jeden krok PSO algoritmu.

Dalším možným přístupem ve spolupracujících PSO algoritmech je konkurenční PSO (CONPSO). Tento algoritmus používá dvou hejn, která konkurenčně prohledávají prostor řešení. Tato dvě hejna si však často vyměňují informaci o nejlepší nalezené pozici. Tyto přístupy lze také kombinovat. Například vytvořením algoritmu, který

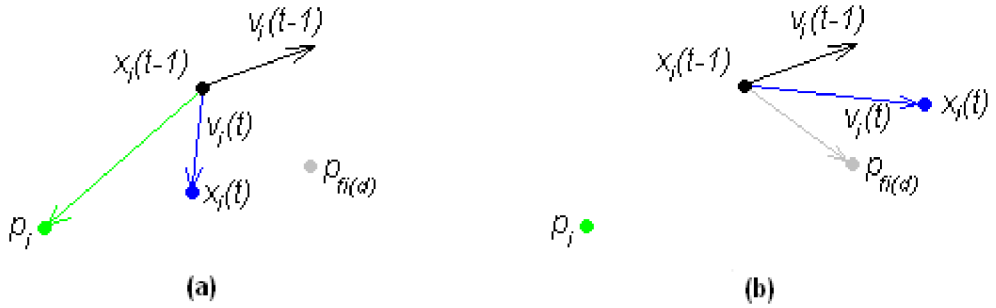
používá dvě konkurenční hejna, která prohledávají prostor pomocí algoritmu CPSO-S nebo CPSO-H [19].

PSO s úplným učením (CLPSO)

V této úpravě PSO algoritmu je rovnice pro aktualizaci rychlosti částice změněna na:

$$v_i^d(t) = \varphi_{ic} \cdot v_i^d(t-1) + \varphi \cdot rand_1 \cdot \left(pbest_{fi(d)}^d - x_i^d(t-1)\right), \qquad (6\text{-}12)$$

kde d je index rozměru a $f_i(d)$ definuje které částice p_{best} má částice i následovat.



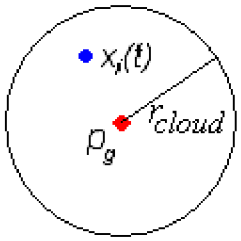
obrázek 6-2:Aktualizace pozice částice u algoritmu CLPSO.

Pro každou dimenzi d částice i je vygenerováno náhodné číslo. Pokud je toto číslo větší než hodnota pravděpodobnosti učení této částice (Pc_i), pak bude částice následovat své vlastní p_{best}, viz obrázek 6-2 (a). Jinak bude následovat p_{best} jiné částice, viz obrázek 6-2 (b). Tato částice se určí turnajem, a to tak, že se nejprve náhodně vyberou dvě částice z hejna a použije se ta částice, jež má větší hodnotu p_{best}. Aby bylo zajištěno, že se částice učí jen z dobrých příkladů a nesměřují špatným směrem, je učení umožněno až po předem daném počtu iterací algoritmu m.

Parametry φ, Pc a m se v algoritmu CLPSO musí vyladit. Parametr Pc je vhodné nastavit pro každou částici jinak a tím zajistit, že částice mají různou schopnost detailního prohledávání. Výhodou tohoto přístupu je, že všechny částice jsou potencionálními vůdci hejna. Tím je snížena šance uváznutí v lokálním optimu. Navíc je zvýšena rozmanitost populace díky tomu, že v každém rozměru je jako vzor použita jiná částice [11].

Kvantové PSO (QSO)

Kvantová částicová optimalizace vychází z atomového modelu. V tomto modelu jsou však orbity elektronů nahrazeny kvantovým mračnem. Jedná se v podstatě o pravděpodobnostní distribuční funkci určující, kde se během měření elektron nalézá. Měření můžeme nazývat evaluací. Kvantové částice se potom náhodně nalézají v hyperkouli o poloměru r_{cloud}, se středem v globálním optimu p_g.



obrázek 6-3: Aktualizace pozice částice algoritmu QSO.

Rychlost těchto částic se nebere v úvahu a rovnice aktualizace pozice částice je velmi jednoduchá:

$$\vec{x}_i(t) \in B(r_{cloud}). \tag{6-13}$$

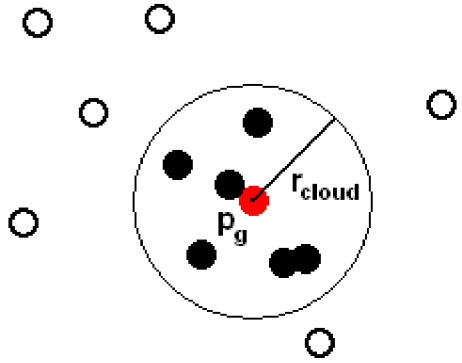
Tyto částice nejsou odpuzovány od ostatních částic ani nejsou ničím přitahovány. Proto se tento algoritmus nepoužívá samostatně, ale slouží pro podrobné prohledávání prostoru kolem globálního optima, nalezeného jinou variantou PSO algoritmu.

Kombinace PSO a QSO (QPSO)

Jak bylo uvedeno v předchozí podkapitole, kvantová optimalizace na bázi částicových hejn se obvykle nepoužívá samostatně, ale jako doplněk jiného prohledávacího algoritmu PSO. Využití QSO může být dvojí. Buď lze QSO použít pro podrobné prohledávání okolí nejlepšího řešení nalezeného klasickým PSO, či některou z jeho variant. V této aplikaci je nejprve nalezeno hrubé optimum a poté se kolem něj vytvoří kvantové mračno o malém poloměru r_{cloud}, které může optimum nalézt přesněji.

Druhou možností jeho využití je sledování optima v dynamických úlohách. V tomto případě algoritmus pracuje tak, že je hejno rozděleno na dvě části (nemusí být stejné velikosti). Jedna část je tvořena normálními částicemi, tj. částicemi chovajícími se dle rovnic (5-5) a (5-1). Druhá část je tvořena kvantovými částicemi. Tyto hejna pracují souběžně, normální částice hledají optimum funkce a kvantové částice sledují

jeho pohyb v čase. Pokud se optimum pohne o menší vzdálenost než je r_{cloud}, je pravděpodobné, že toto nové řešení bude pokryto kvantovým mračnem. To vede k tomu, že tento algoritmus najde nové řešení v dynamických úlohách rychleji než jiné varianty PSO algoritmu (Blackwell8).



obrázek 6-4: Algoritmus QPSO (plné kruhy představují kvantové částice, prázdné kružnice představují normální částice).

7 Řešený problém

Řešeným problémem je optimalizace cest obchodních zástupců a servisních techniků společnosti INVEA-TECH k zákazníkům. Je to typický příklad TSP problému. My však budeme řešit obecnější problém, jelikož máme k dispozici více vozidel. Pokud by bylo vozidlo jen jedno, jedná se o běžný TSP problém, pokud je jich však více, je nutno úlohu rozložit na několik dílčích TSP problémů. Tento rozklad provedeme shlukovou analýzou, kdy počet shluků je stejný jako počet vozidel.

obrázek 7-1: Mapa zákazníků společnosti INVEA-TECH

Existuje předpoklad (nejen u tohoto konkrétního příkladu), že vozidla budou vyjíždět z nějakého centrálního místa, kde vozidla parkují. Tato cesta z centrály není v příkladu uvažována. V podstatě by to však znamenalo, že jeden bod by byl společný všem shlukům. Vozidla vyjedou z centrály, navštíví místa přidělená shlukovou analýzou v pořadí určeném řešením TSP problému a opět se vrátí.

7.1 Algoritmus PSO pro shlukování

Algoritmus PSO pro shlukovou analýzu je klasickým PSO algoritmem. Jediné čím se liší je struktura částic a fitness funkce. Informace o každé částici se skládá z osmi

vektorů. Jsou to x a y pozice částice, rychlost částice v x a y směru, dále hodnota fitness funkce pro danou pozici částice a nakonec nejlepší známá x a y pozici částice a hodnota fitness funkce na této pozici (p_{best}). Délka vektorů je rovna počtu shluků, tedy k. Tedy pozice částice se skládá z k pozic středů shluků a každý střed shluku má svůj vektor rychlosti.

Fitness funkce je pak klasická chyba shlukování, jak ji známe například z metody k-means, rovnice (4-6). Každá částice roje tak představuje jednu možnou konfiguraci fiktivních středů shluků.

Algoritmus:

1. Inicializace hejna. Každé částici jsou náhodně přiřazeny pozice středů shluků v prohledávaném prostoru a náhodný vektor rychlosti.
2. Pro každou částici je vypočítána hodnota fitness funkce. Tj. pro každou částici jsou prvky rozděleny do shluků a vypočítána současná chyba shlukování E.
3. Porovnání současné hodnoty fitness funkce částice s její p_{best}. Pokud je současná hodnota lepší, je označena za p_{best} a do p_i je uložena současná poloha částice.
4. Nalezení částice s nejlepší fitness funkcí. Tato hodnota je označena za g_{best} a její poloha za p_g.
5. Aktualizace pozic a rychlostí částic dle rovnic (5-1) a (5-6).
6. Opakování kroků 2-5 dokud nejsou splněny podmínky ukončení. Tedy dokud není dosažen maximální počet iterací algoritmu. Poloha částice p_g pak udává pozice fiktivních středů shluků, pro něž je chyba shlukování nejmenší.

Tento algoritmus je tak kombinací klasického algoritmu PSO (s použitím konstanty setrvačnosti) a shlukové metody k-means.

7.2 Algoritmus PSO pro TSP

Algoritmus PSO pro řešení problému obchodního cestujícího je v mnoha ohledech složitější než algoritmus předchozí. Základním rozdílem je, že využívá diskrétní PSO. Jako všechny varianty PSO, i diskrétní PSO vyžaduje nadefinovat několik základních složek. Jsou jimi stavový prostor přípustných řešení, pozice částice, rychlost částice a její pohyb a v neposlední řadě účelová funkce.

Pozice částice a stavový prostor

Nechť $G = \{E_G, V_G\}$ je ohodnocený graf, ve kterém hledáme Hamiltonovskou kružnici. E_G je množina ohodnocených hran a V_G je množina vrcholů. Vrcholy grafu jsou očíslovány od 1 do N, takže každý element množiny V_G můžeme označit indexem i, $i \in (1,2,\ldots,N)$. Každý prvek množiny E je trojice $(i, j, w_{i,j}), i, j \in V_G, w_{i,j} \in R^+$. Jelikož hledáme cyklus, uvažujme sekvenci N vrcholů, kde každý vrchol je jiný. Takovouto sekvenci vrcholů označíme N-cyklus a nazveme jej pozice částice x. A stavovým prostorem pak rozumíme konečnou množinu všech N-cyklů.

Účelová funkce

Podle předchozí definice je pozice částice $x = (n_1, n_2, \ldots, n_N), n_i \in V_G$, N-cyklus. Tento cyklus je přípustný pouze pokud existují všechny hrany (n_i, n_{i+1}). To znamená, že graf musí být úplný. Pokud tomu tak není, musíme vytvořit fiktivní hranu, jejíž ohodnocení w bude natolik vysoké, abychom měli jistotu, že tato virtuální hrana nikdy nebude součástí optimálního řešení. Po zavedení tohoto opatření mají všechny hrany grafu své ohodnocení a pak může definovat účelovou funkci jako

$$f(x) = \sum_{i=1}^{N-1} w_{n_i, n_{i+1}} + w_{n_1, n_N} . \tag{7-1}$$

Tato účelová funkce má konečný počet hodnot a její globální minimum odpovídá nejlepšímu řešení TSP problému.

Rychlost částice a její pohyb

Nyní musíme definovat operátor rychlosti v, což je pravděpodobně nejsložitější část algoritmu. Pozice částice (N-cyklus) je vlastně permutací N vrcholů grafu G. Operátor rychlosti pak bude provádět prohození prvků v této permutaci tak, že po jeho provedení dostaneme jinou pozici částice. Rychlost v je potom seznam dvojic vrcholů, které budou prohozeny a $|v|$ je délka tohoto seznamu.

$$v = ((i_1, j_1), (i_1, j_1), \ldots, (i_{|v|}, j_{|v|})), \quad i_k, j_k \in V_G . \tag{7-2}$$

Zkráceně budeme rychlost v zapisovat jako $v = ((i_k, j_k))$, což znamená prohození vrcholů (i_1, j_1), potom vrcholů (i_2, j_2) atd. až na nakonec vrcholů $(i_{|v|}, j_{|v|})$. Operace

sčítání rychlostí částice $v_1 + v_2$ pak chápeme jako konkatenaci dvou seznamů. Například $v_1 = ((1,3),(2,5))$, $v_2 = (3,5)$, pak $v = v_1 + v_2 = ((1,3),(2,5),(3,5))$.

Pohybem částice rozumíme aktualizaci pozice částice dle rovnice (5-1). V našem případě to znamená, že provedením sekvence prohození vrcholů v N-cyklu x dostaneme N-cyklus x'. Například mějme pozici částice $x = (2,5,4,1,3)$ a rychlost $v = ((2,1),(1,4))$. Pak nová pozice částice bude $x' = (4,5,1,2,3)$.

Poslední operací nutnou pro algoritmus PSO pro řešení TSP je rozdíl dvou pozic částic. Tedy operátor opačný rychlosti [15]. Jinak řečeno, výsledkem rozdílu dvou pozic je rychlost x_2-$x_1 = v$ a pohybem částice z pozice x_1 rychlostí v se dostaneme do pozice x_2, x_1+$v = x_2$. Například pozice $x_1 = (2,5,4,3,1)$ a pozice $x_2 = (2,4,3,5,1)$, pak rychlost v = x_2-$x_1 = ((5,3),(3,4))$.

Algoritmus

Samotný algoritmus PSO je velmi podobný tomu, jak jsme si jej představili již dříve. Rovnice aktualizace pozice částice je v podstatě stejná jako u klasické varianty, ale rovnice aktualizace rychlosti částice se změní na

$$\vec{v}_i(t) = \vec{v}_i(t-1) + \vec{\alpha} \cdot (\vec{p}_i - \vec{x}_i(t-1)) + \vec{\beta} \cdot (\vec{p}_g - \vec{x}_i(t-1)). \qquad (7\text{-}3)$$

Jak je patrno, jedinou změnou jsou parametry $\vec{\alpha}$ a $\vec{\beta}$. Jsou však změnou velmi podstatnou. Jsou to totiž vektory náhodných čísel v rozmezí $\langle 0\text{-}1 \rangle$. Tyto vektory mají stejný počet prvků jako je délka rychlosti mezi p_i a x, respektive mezi p_g a x. Pokud je náhodné číslo pro dané prohození dvojice vrcholů nižší než zadaný práh, toto prohození se při aktualizaci rychlosti částice neuvažuje. To zabezpečuje, že se neprovedou všechna prohození rozdílných vrcholů částice s její nejlepší pozicí, nebo nejlepší globální pozicí, a tudíž se pozice částice pouze přibližuje nejlepším známým řešením a neproměňuje se v ně. Tím je zaručena diverzita populace.

Problémem však může být, že počet prohození dvojic vrcholů neúměrně stoupá. Uvědomme si, že v každém kroku je zachována předchozí rychlost (počet prohození) a k ní jsou neustále přidávány další dvě sekvence záměn vrcholů. To by mohlo vést k neúnosným výpočetním časům. Proto je vhodné omezit maximální počet prohození, tedy maximální délku seznamu $|v|$ parametrem v_{max}. Zde by opět mohl nastat problém, protože bychom mohli zachovávat pouze rychlost z předešlého kroku a neaktualizovat

pohyb směrem k nejlepším řešením. Z tohoto důvodu je lepší omezovat parametrem v_{max}, každou část rovnice (7-3) zvlášť. Tím je implementována i jistým způsobem modifikovaná konstanta setrvačnosti, protože do dalšího kroku algoritmu se nepředává celá rychlost částice v, ale jen její část.

Algoritmus PSO pro TSP pak vypadá následovně:

1. Inicializace hejna. Každé částici je přiřazena náhodná permutace vrcholů grafu a náhodný seznam dvojic vrcholů k prohození (rychlost v) o délce $v_{max.}$
2. Pro každou částici je vypočítána hodnota fitness funkce dle rovnice (7-1).
3. Porovnání současné hodnoty fitness funkce částice s její p_{best}. Pokud je současná hodnota lepší, je označena za p_{best} a do p_i je uložena současná poloha částice.
4. Nalezení částice s nejlepší fitness funkcí. Tato hodnota je označena za g_{best} a její poloha za p_g.
5. Aktualizace rychlostí částic dle rovnice (7-3) a pozic částic dle rovnice (5-1).
6. Výpočet rozdílu nových a starých pozic částic, tzn. nových rychlostí částic a jejich omezení na délku $v_{max.}$
7. Opakování kroků 2-6 dokud nejsou splněny podmínky ukončení. Tedy dokud není dosažen maximální počet iterací algoritmu. Poloha částice p_g pak udává N-cyklus vrcholů grafu s nekratší cestou a g_{best} udává její délku.

7.3 Struktura programu

Program byl implementován v prostředí Matlab 7.1 a skládá ze z těchto zdrojových souborů:

gui.m, gui.fig

Zde je implementováno grafické uživatelské rozhraní programu. Toto prostředí bylo vytvořeno pomocí nástroje Matlab GUI design environment (GUIDE), viz obrázek 7-2.

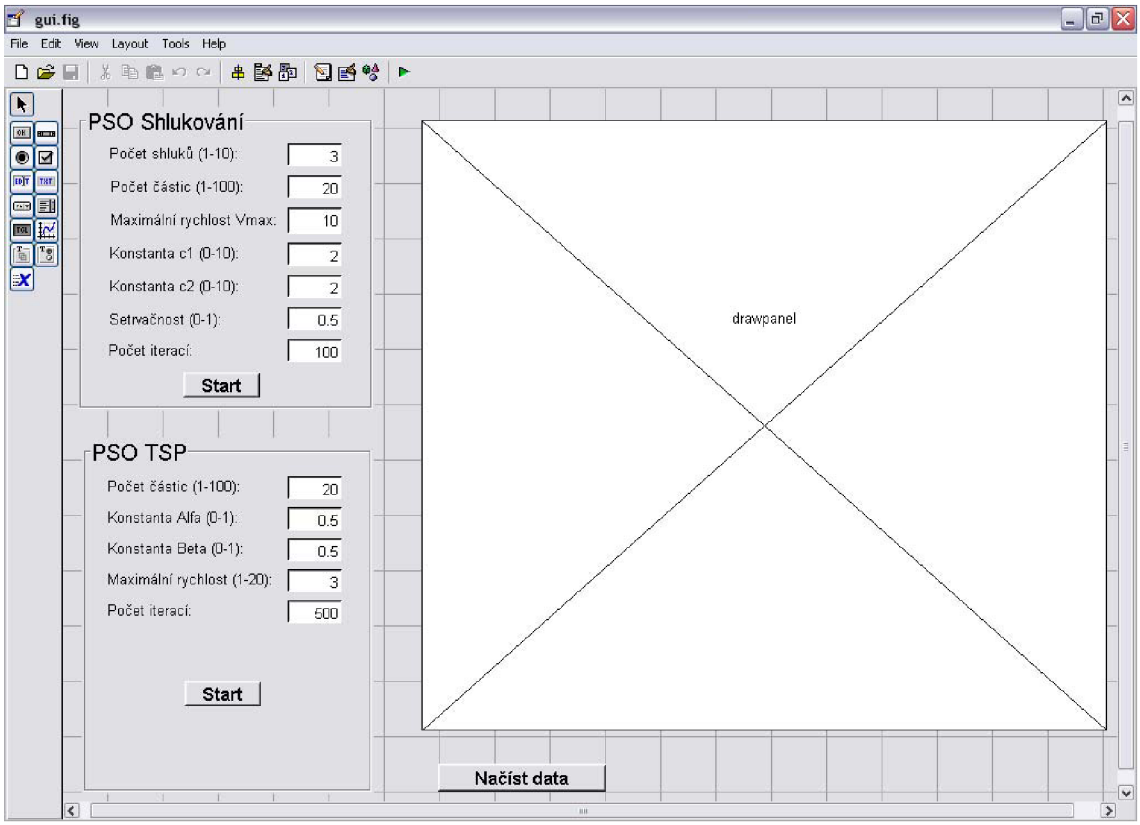
Dále jsou zde implementovány funkce pro načtení dat z xls souboru, načtení parametrů simulace a také kód, který rozděluje načtená data pro TSP podle vytvořených shluků. Samozřejmě je zde také implementováno vykreslení výsledků simulace.

clusteringPSO.m

V tomto souboru, jak název napovídá, je implementován algoritmus PSO pro shlukovou analýzu. Výstupem tohoto souboru jsou souřadnice středů shluků.

calculate.m

Toto je pomocný soubor, kde je implementován výpočet fitness funkce algoritmu PSO pro shlukovou analýzu. Tzn. suma vzdáleností bodů od středů shluků.



obrázek 7-2: Nástroj GUIDE

TSP_PSO.m

Implementace algoritmu PSO pro TSP, vrací optimální cestu mezi zadanými městy. Tedy mezi městy patřících do jednoho shluku.

calculate2.m

Opět pomocný soubor s implementovaným výpočtem fitness funkce, tentokrát algoritmu PSO pro TSP. Vrací délku cesty vstupní permutace měst.

Formát vstupních dat

Jak již bylo zmíněno, program pracuje s daty ve formátu xls. Je však nutno ukázat jak přesně tato data mají vypadat, aby byl program schopen zpracovat. Nejdůležitějším faktem je, že se data skládají ze dvou částí, to znamená, že jsou data umístěna na dvou samostatných listech v jednom xls souboru. První část se nachází na listu pojmenovaném „Lokace“ a jedná se o souřadnice měst (bodů). V našem případě to jsou GPS souřadnice jednotlivých měst ČR, v kterých se v současné době nacházejí sídla zákazníků společnosti INVEA-TECH, převedené na stupně. Tato část dat je potřebná pro vykreslení jednotlivých měst a pro shlukovou analýzu.

	A	B	C
1	Brno	16,6087	49,2032
2	Česká lípa	14,5407	50,6837
3	Č. Budějovice	14,4779	48,9738
4	Děčín	14,2039	50,7751
5	Frýdek-Místek	18,3422	49,6799
6	Havířov	18,4309	49,7808
7	Havl. Brod	15,5802	49,6068
8	Hodonín	17,1174	48,8519
9	Hr. Králové	15,8315	50,2108
10	Chomutov	13,4050	50,4583
11	Jablonec n. Nisou	15,1623	50,7225
12	Jihlava	15,5848	49,3972
13	Karlovy Vary	12,8812	50,2336

obrázek 7-3: Formát dat listu Lokace

Druhá část dat je uložena na listu „Vzdalenosti“ a obsahuje tabulku vzdáleností mezi jednotlivými místy. Tato data jsou nutná pro část programu řešící TSP problém. Je nutné si uvědomit, že zadaná vzdálenost je silniční. Proto nelze tyto vzdálenosti odvodit přímo v programu ze souřadnic míst. Pro zjednodušení lze uvažovat i nejkratší vzdálenosti mezi dvěma body, tzn. letecké. Jedná se o obyčejný výpočet Euklidovské vzdálenosti dle vzorce (4-1), což hravě zvládne vypočítat program Microsoft Excel, či jiný tabulkový procesor, ze zadaných souřadnic bodů.

	A	B	C	D	E	F	G	H	I
1		Brno	Česká lípa	Č. Budějov	Děčín	Frýdek-Mís	Havířov	Havl. Brod	Hodonín
2	Brno	99999	258	216	312	160	174	105	70
3	Česká lípa	258	99999	223	36	357	358	152	321
4	Č. Budějovice	216	223	99999	248	372	387	127	279
5	Děčín	312	36	248	99999	393	394	189	374
6	Frýdek-Místek	160	357	372	393	99999	15	262	166
7	Havířov	174	358	387	394	15	99999	276	180
8	Havl. Brod	105	152	127	189	262	276	99999	168
9	Hodonín	70	321	279	374	166	180	168	99999
10	Hr. Králové	142	126	209	163	230	233	82	211
11	Chomutov	297	102	227	78	453	468	208	360
12	Jablonec n. Nisou	256	64	221	84	319	321	150	319
13	Jihlava	89	178	123	232	245	260	26	152
14	Karlovy Vary	330	151	211	127	486	501	241	393

obrázek 7-4: Formát dal listu Vzdalenosti

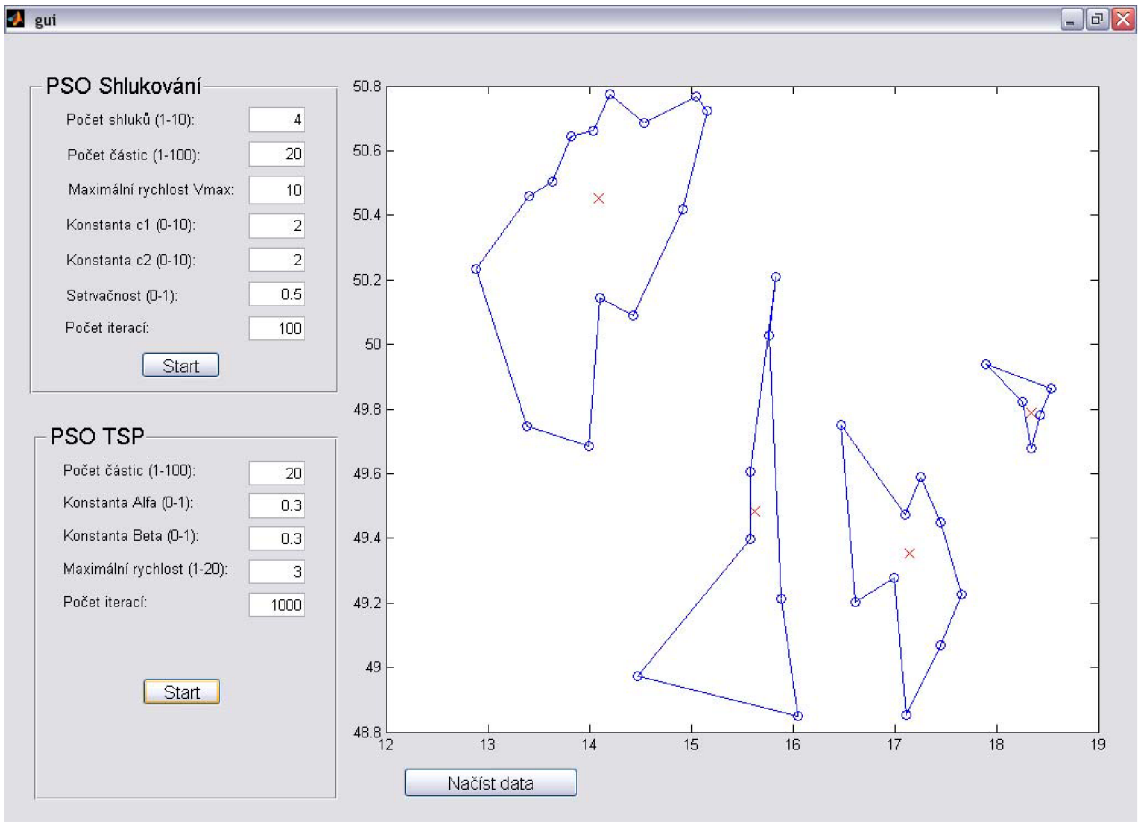
Jak bylo řečeno v kapitole 7.2, všechny hrany grafu musejí být ohodnoceny a to i ty, kde ve skutečnosti spojnice mezi dvěma místy neexistuje. V takovém případě musí být dosazena dostatečně vysoká hodnota, aby tato spojnice nikdy nebyla vybrána v optimálním řešení. Je vhodné takto také ohodnotit vzdálenost z bodu do něj samotného, aby se náhodou v optimální cestě nenacházely smyčky zacyklené na jednom místě. V našem případě byla tato hodnota zvolena jako číslo 99999, ale velikost tohoto čísla záleží na ostatních vzdálenostech a je plně v režii uživatele.

7.4 Ovládání programu

Okno programu (viz obrázek 4-1) si lze pro jednodušší popis ovládání rozdělit na tři části. V první části je vykreslovací plocha programu a tlačítko pro načtení dat, se kterými se bude provádět výpočet. Tyto data musí být ve správném formátu, tak jak bylo popsáno v předchozí kapitole. Po načtení dat se zobrazí jednotlivá místa jako modrá kolečka.

Druhá část je označena jako „PSO Shlukování“ a slouží pro zadání parametrů PSO algoritmu řešícího problém shlukování. Za názvem parametru téměř vždy následuje závorka, ve které je uveden doporučený rozsah hodnot daného parametru. Nastavitelnými parametry jsou počet shluků, počet částic hejna, maximální rychlost částic a konstanty c_1, c_2 a konstanta setrvačnosti. Pro úplnost, parametr c_1 představuje přitažlivost k nejlepší pozici částice a parametr c_2 přitažlivost k nejlepší částici. Konstanta setrvačnosti se zadává jako desetinné číslo v anglickém formátu, tj. oddělené tečkou. Posledním parametrem je počet kroků (iterací) simulace. Po stisknutí tlačítka

Start v této části okna programu se provede výpočet shlukové analýzy a do okna pro zobrazení výsledků se vykreslí červené křížky znázorňující středy shluků.



obrázek 7-5: Uživatelské rozhraní programu

Třetí a poslední část okna programu nese název „PSO TSP“ a je určena pro zadání parametrů PSO algoritmu řešícího problém obchodního cestujícího. Těmito parametry jsou počet částic hejna, konstanty Alfa a Beta, maximální rychlost částic a počet iterací algoritmu. Konstanty Alfa a Beta představují práh přitažlivosti k nejlepší pozici částice a k nejlepší částici. Musí být zadány jako desetinné číslo v rozmezí ⟨0-1⟩, kde 0 značí, že částice k těmto pozicím není přitahována vůbec (tj. neprovádí se žádné prohazování vrcholů) a 1 představuje maximální přitažlivost (tj. provádí se všechny záměny vrcholů, až do maximální délky seznamu dvojic vrcholů neboli maximální rychlosti částice). Po stisknutí tlačítka Start v této části okna programu se provede výpočet TSP problému a v okně pro zobrazení výsledků se vykreslí modré spojnice míst.

8 Testy a dosažené výsledky

Při testování optimalizačních algoritmů nás většinou zajímá časová závislost výpočtu na velikosti vstupu. Lepší než zjišťovat dobu výpočtu jednotlivých případů, je zjišťovat počet iterací algoritmu. Tento ukazatel je mnohem univerzálnější a není závislý na hardwaru, na kterém testy probíhají. Z tohoto důvodu je v následujících testech vždy udán počet iterací, při kterých algoritmus dosáhl optima. Samozřejmě, jako u všech optimalizačních metod pracujících s faktorem náhody, je při každém běhu programu tento výsledek jiný. Proto jsou výsledné hodnoty v tabulkách vždy průměrem z deseti běhů programu.

Shlukování

Při shlukové analýze pomocí algoritmu PSO nemusíme vždy dosáhnout optimálních pozic středů shluků, ale vždy dosáhneme optimálního rozdělení prvků do shluků. Proto nemusíme hledat optimální řešení, ale stačí nám řešení dostatečně kvalitní. Při testování shlukové analýzy zkoumáme dobu výpočtu vzhledem k počtu částic a vzhledem k počtu shluků.

Pro směrodatné výsledky testů, je také správné nastavení parametrů algoritmu PSO. Podrobně se touto problematikou zabývá kapitola 5.2. Parametry přitažlivosti k nejlepší známé pozici částice a k nejlepší částici c_1 a c_2, byly nastaveny dle výše popsaných doporučení. Hodnota maximální rychlosti nemá v tomto případě na výpočet příliš velký vliv, slouží pouze k tomu, aby částice neprohledávaly zbytečně velký prostor řešení. Jeho velikost se určí podle rozsahu zadaných hodnot. Počet částic je dle experimentálních výsledků vhodné nastavovat v rozmezí 20-50.

tabulka 8-1: Nastavení parametrů testu shlukové analýzy

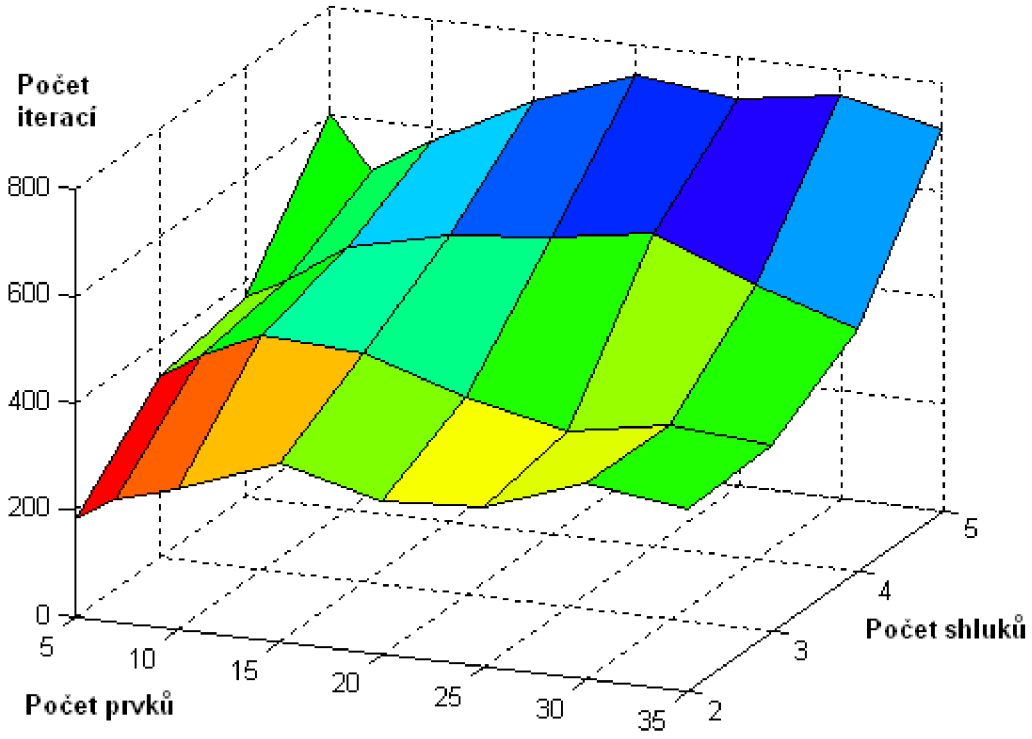
Parametr	Hodnota	Popis
n	20	Počet částic.
Vmax	1	Maximální rychlost částice.
c1	2	Přitažlivost k nejlepší pozici částice.
c2	2	Přitažlivost k nejlepší částici.
setrvačnost	0.5	Snaha pokračovat stále stejným směrem.

Potřebný počet iterací je předmětem zkoumání. Jak je z následujících výsledků zřejmé, jeho hodnota nepřekročí hranici tisíc iterací.

tabulka 8-2: Výsledky testu shlukové analýzy

Počet měst **Počet shluků**	**5**	7	**10**	**15**	**20**	**25**	**30**	**35**
2	185	229	264	334	286	299	367	342
3	336	388	438	427	367	327	363	348
4	373	412	490	535	553	584	513	453
5	603	504	573	674	745	723	754	715

Z výsledků je patrné, že rostoucí počet prvků nemá na dobu výpočtu příliš velký vliv. Mnohem zásadnější je vliv počtu shluků, do kterých prvky rozdělujeme. Kvalitního řešení shlukové analýzy jsme ve všech testovaných případech dosáhli do 750 iterací algoritmu. Paradoxně nejproblémovějším případem je, pokud je počet částic stejný jako je počet shluků.



obrázek 8-1: Výsledky testu shlukové analýzy

Z grafu je zřetelné, že s rostoucím počtem prvků se doba výpočtu zvyšuje jen mírně, zato vliv počtu shluků je velmi velký. Velký vliv na dobu výpočtu shlukové analýzy má také vliv kvalita prvků, tj. jak dobré shluky prvky tvoří. Tento vliv však nebyl testován, neboť program bude využíván pro rozdělování měst České Republiky, či jiných míst. Takováto data obecně dobré shluky netvoří.

Problém obchodního cestujícího

Při testování algoritmu PSO pro řešení problému obchodního cestujícího zkoumáme závislost doby výpočtu na počtu měst a také zkoumáme vliv jednoho parametru algoritmu. Konkrétně je tímto parametrem maximální rychlost částic, ta má totiž u diskrétní varianty algoritmu PSO mnohem větší vliv než u varianty spojité. Pokud je maximální rychlost částic vysoká, algoritmus konverguje rychleji, ale nemusí vždy nalézt správné řešení. Pokud je naopak maximální rychlost nízká, správné řešení je nalezeno vždy, ale výpočet trvá delší dobu. To je zohledněno i v následující tabulce, kde konfigurace kdy nebylo vždy dosaženo optimálního řešení v rozumném čase, jsou označeny hvězdičkou. Toto je neduh většiny „chytrých algoritmů“, kdy je nutno pro nalezení optimálního nebo alespoň suboptimálního řešení provést několik běhů výpočtu.

Pro správné výsledky testů je opět nutné definovat správné nastavení parametrů algoritmu PSO pro řešení problému TSP. Počet částic byl zvolen stejný, jako u shlukování. Nutno podotknout, že pro řešení úlohy s větším počtem měst je vhodné nastavit větší počet částic. Pro zachování jednotnosti výsledků, byl však ponechán stále stejný počet částic, ale test proběhl dvakrát. Jednou pro dvacet částic, podruhé pro padesát. Parametry α a β byly nastaveny dle literatury [12]. Nastavení parametru maximální rychlosti částic je předmětem zkoumání.

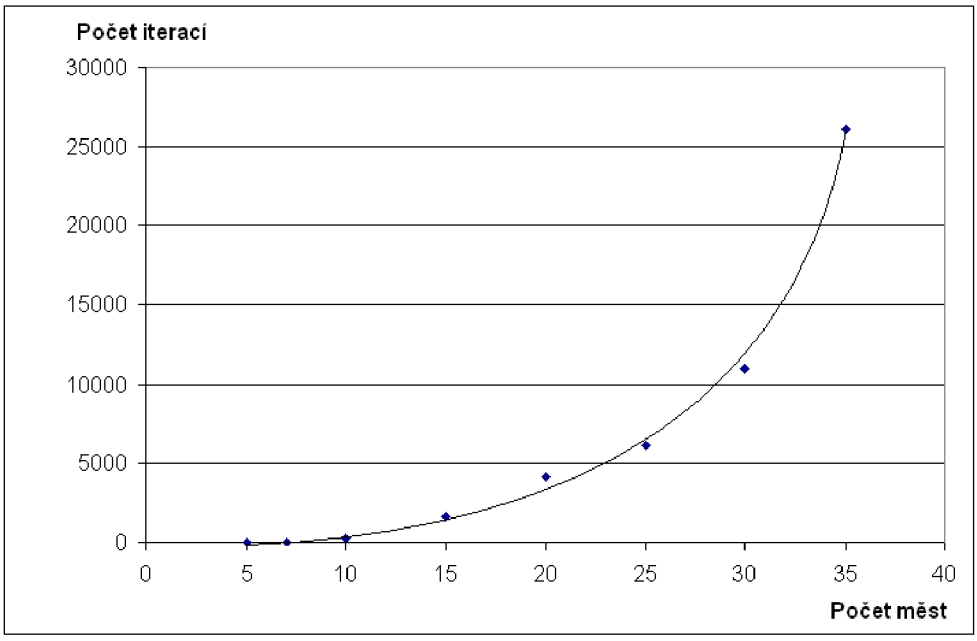
tabulka 8-3: Nastavení parametrů testu PSO algoritmu pro TSP

Parametr	Hodnota	Popis
n	20/50	Počet částic.
α	0.3	Práh přitažlivosti k nejlepší pozici částice.
β	0.3	Práh přitažlivosti k nejlepší částici.

tabulka 8-4: Výsledky testu PSO algoritmu pro TSP pro 20 částic

Počet měst **Max. rychlost**	**5**	7	**10**	**15**	**20**	**25**	**30**	**35**
1	1	19	166	1225	3844	6466	10897	34681
3	1	13	295	1635	4122	6171	10979	26125
5	1	21	488	2090	3261*	6166*	9342*	21324*
10	1	15	466*	1050*	2160*	4857*	6831*	8125*

Výsledky testů ve velké míře potvrdili předpoklady, že při vyšší maximální rychlosti částic, má algoritmus problémy s nalezením optimálního řešení. Zajímavým zjištěním je, že při malém počtu měst algoritmus nalezne řešení rychleji při nízké maximální rychlosti. To je pravděpodobně způsobeno malým prostorem řešení, který pomalejší částice snadněji prohledají. Obecně nejvýhodnějším nastavením parametru V_{max} je hodnota 3. Pro tuto hodnotu byl sestaven následující graf.



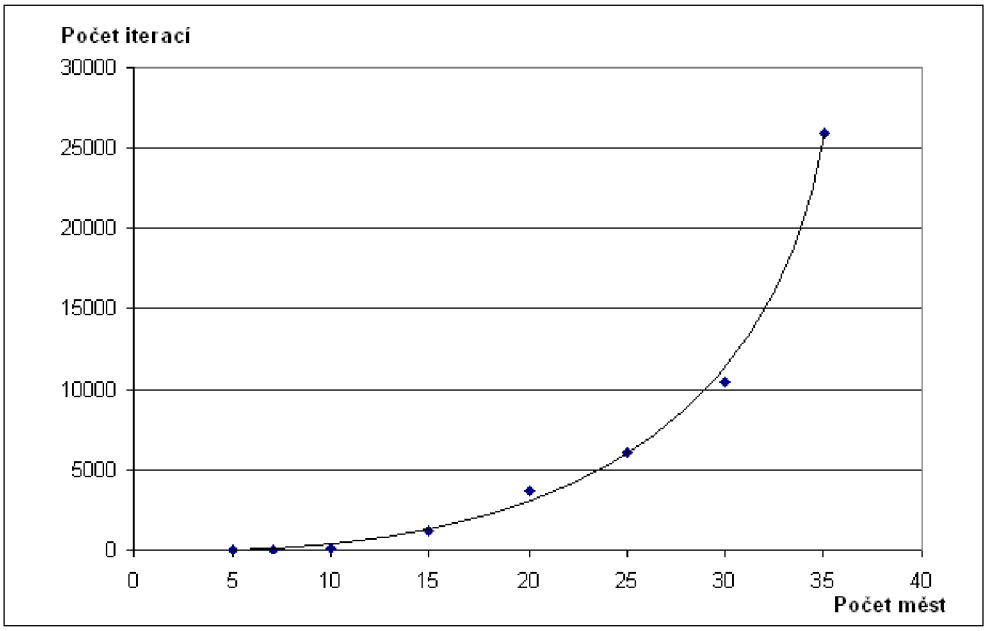
obrázek 8-2: Výsledky testu PSO algoritmu pro TSP pro Vmax = 3, n = 20

Z tohoto grafu je zřejmé, že doba výpočtu algoritmu se s rostoucím počtem měst rychle zvyšuje. Stále však roste o hodně pomaleji, než kdybychom problém obchodního cestujícího řešili „hrubou silou“.

tabulka 8-5: Výsledky testu PSO algoritmu pro TSP pro 50 částic

Počet měst **Max. rychlost**	**5**	7	**10**	**15**	**20**	**25**	**30**	**35**
1	1	10	161	1033	2799	7464	11340	38797
3	1	3	140	1185	3733	6124	10531	25916
5	1	5	416	1782	3415*	5814*	8647*	19943*
10	1	8	334*	1164*	1687*	4461*	5617*	7651*

Výsledky tohoto testu jsou obdobné jako u předchozího případu. Potvrdilo se, že když je počet částic větší snížil se počet iterací výpočtu, ale ne jeho doba. Protože je v každé iteraci potřeba ohodnotit více částic, prodloužila se doba výpočtu jedné iterace. Takže celkově k úspoře času nedochází. Opět se jako nejvýhodnější velikost maximální rychlosti částice jeví hodnota 3.



obrázek 8-3: Výsledky testu PSO algoritmu pro TSP pro Vmax = 3, n = 50

Algoritmus pro řešení TSP problému pracuje dostatečně rychle a přesně pro menší počty měst. Pokud je měst více, doba výpočtu se značně prodlužuje, ale stále ne tak moc jako u některých jiných metod. Problémem je, že ne vždy tato metoda nalezne optimální řešení, ale nalezne alespoň řešení poměrně kvalitní. Pokud je požadováno skutečně optimální řešení je nutno běh programu několikrát opakovat.

9 Závěr

Diplomová práce se zabývala implementací algoritmů řešících problém obchodního cestujícího a problém shlukování. Spojením těchto dvou algoritmů dohromady vznikl program, který řeší problém obchodního cestujícího pro více vozů, neboli řešení problému více obchodníků cestujících zároveň. V práci byly nejprve tyto optimalizační problémy podrobně popsány a pak byly navrženy algoritmy založené na optimalizace na bázi částicových hejn (PSO). Tím také bylo ukázáno, že optimalizační metoda PSO je obecně použitelná na širokou řadu optimalizačních problémů.

Navržené algoritmy byly implementovány v softwarovém nástroji Matlab, kde také byly důkladně testovány. Tím bylo zjištěno optimální nastavení parametrů algoritmu, aby bylo následné řešení zadaných problémů co nejefektivnější. Vytvořený program je tak obecně využitelný všude tam, kde je nutná optimalizace logistických tras, například rozvážka materiálu, zboží, pošty či peněz nebo objíždění zákazníků.

Díky vytvořenému programu mohou obchodní zástupci či servisní technici společnosti INVEA-TECH cestovat k zákazníkům mnohem efektivněji než tomu bylo doposud. Tím dochází jak k úspoře cestovních nákladů, tak i k rychlejšímu obsloužení zákazníků. A spokojený zákazník je to nejdůležitější co firma může mít.

Literatura

[1] *Artificial bee colony algorithm* – Wikipedia, The Free Encyclopedia. [online]. 2008. [cit. 2009-12-20]. Dostupné z <http://en.wikipedia.org/wiki/Artificial_Bee_Colony_Algorithm>

[2] van den BERGH, F. *An analysis of particle swarm optimizers*. (disertační práce) Pretoria: University of Pretoria, 2001. 283 s.

[3] DAVIS, L. *Handbook of Genetic Algorithms*. USA: Int. Thomson Com. Press, 1991. 385s. ISBN 1-850-32825-0.

[4] DOSTÁL, P. *Advanced Economic Analyses*. Brno: CERM, 2008. 80s. ISBN 978-80-214-3564-3.

[5] DOSTÁL, P. *Pokročilé metody analýz a modelování v podnikatelství a veřejné správě*. Brno: Cerm, 2008. 340 s. ISBN 978-80-7204-605-8

[6] HLINĚNÝ, P *Optimalizační úlohy*. Výukový text do předmětu Úlohy lineární a celočíselné optimalizace a jejich řešení. Brno: FI MU v Brně, [online]. 2007. [cit. 2010-4-7]. Dostupné z <www.fi.muni.cz/~hlineny/Teaching/OU/OU-text07.pdf>

[7] HULA, T. *Experimenty s rojovou inteligencí (swarm intelligence)*. (diplomová práce) Brno: FIT VUT v Brně, 2008. 98 s.

[8] INVEA-TECH. Webové stránky společnosti [online], [cit. 2010-5-7]. Dostupné z <http://www.invea.cz/spolecnost>

[9] KARABOGA, D. a kol. *Artificial bee colony (ABC) algorithm*. [online]. 2008. [cit. 2009-12-20]. Dostupné z <http://mf.erciyes.edu.tr/abc/>

[10] KELBEL, J., ŠILHÁN, D. *Shluková analýza.* [online]. [cit. 2010-3-7]. Dostupné z <http://gerstner.felk.cvut.cz/biolab/X33BMI/slides/KMeans.pdf>

[11] LIANGG, J.J. a kol. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. In *IEEE Transactions on Evolutionary Computation*. sv.10, č. 3, 2006. s. 281-295

[12] LIU, X. a kol. An improved particle swarm optimization for traveling salesman problem. In *Advanced intelligent computing theories and applications - Third international conference on intelligent computing*. Chinese academy of science. Berlin: Springer, 2007. s. 803-811

[13] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. *Umělá inteligence (4)*. Praha: ACADEMIA, 2003. 475s. ISBN 80-200-1044-0.

[14] Ministerstvo spravedlnosti ČR. *Obchodní rejstřík a sbírka listin*. [online]. [cit. 2010-5-20]. Dostupné z <http://www.justice.cz>

[15] ONWUBOLU, G.C., BABU, B.V. *New optimization techniques in engineering*. Berlin: Springer, 2004. 712 s. ISBN: 978-3-540-20167-0

[16] RUDOLFOVÁ, I. *Shluková analýza*. Přednáška z předmětu Získávání znalostí z databází. FIT VUT v Brně, 2008

[17] *Swarm intelligence* – Wikipedia, The Free Encyclopedia. [online]. 2008. [cit. 2009-12-18]. Dostupné z <http://en.wikipedia.org/wiki/Swarm_intelligence>

[18] THE MATHWORKS. MATLAB – *Genetic Algorith Toolbox - User's Guide*. The MathWorks, Inc., 2008

[19] del VALLE, Y. a kol. Particle swarm optimization: Basic concepts, variants and applications in power systems. In *IEEE Transactions on Evolutionary Computation*. sv.12, č. 2, 2008. s. 171-195

[20] ZBOŘIL, F.V. *Strojové učení*. Přednáška z předmětu Základy umělé inteligence. FIT VUT v Brně, 2006

[21] ZIMMER, C. *From ants to people, an instinct to swarm*. International Herald Tribune. [online]. 2007. [cit. 2010-3-10]. Dostupné z <http://www.iht.com/articles/2007/11/13/healthscience/13traff.php>

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

ABC	artificial bee colony - umělá včelí kolonie
ACO	ant colony optimization - optimalizace pomocí mravenčích kolonií
c_1	akcelerační konstanta řídící pohyb k nejlepší pozici částice
c_2	akcelerační konstanta řídící pohyb k pozici nejlepší částice
c_3	akcelerační konstanta řídící pohyb k pozici náhodné částice
C-PSO	composite PSO - kompositní PSO
CA	cellular automata - celulární automat
CCGA	cooperative coevolutionary genetic algorithm - kooperativní koevoluční genetický algoritmus
CLPSO	comprehensive learning PSO - PSO s úplným učením
CONPSO	concurrent PSO - konkurenční PSO
CPSO	cooperative PSO
CPSO-H	CPSO hybrid - hybridní CPSO
CPSO-S	CPSO split - rozdělující CPSO
DEPSO	differential evolution PSO - PSO s diferenciální evolucí
DN-PSO	dynamic neighborhood PSO - PSO s dynamickým sousedstvím
DPSO	dissipative PSO - rozptýlené PSO
EPSO	evolutionary PSO - evoluční PSO
GA-PSO	genetic algorithm PSO - kombinace genetického algoritmu a PSO
g_{best}	hodnota fitness funkce nejlepší částice
GPSO	Gaussian PSO - Gaussovo PSO
MOPSO	multiobjectiv PSO - víceúčelové PSO
mPSO	multiswarm PSO - vícerojové PSO
mQPSO	multiswarm quantum PSO - vícerojové kvantové PSO
mQPSOPC	multiswarm quantum PSOPC - vícerojové kvantové PSOPC
N_{pop}	velikost populace částic
p_{best}	nejlepší hodnota fitness funkce jedné částice
p_g	pozice nejlepší částice (sdílená informace)

p_i	pozice nejlepší hodnoty fitness funkce dané částice (paměť částice)
PSO	particle swarm optimization - optimalizace na bázi částicových hejn
PSOPC	PSO with passive congregation - PSO s pasivním shromažďováním
QPSO	quantum PSO - kombinace PSO a QSO
QSO	quantum swarm optimization - kvantová rojová optimalizace
r_{conv}	poloměr anti-konvergence
r_{cloud}	poloměr kvantového mračna
r_{excl}	poloměr exkluze
SAT	Boolean satisfiability problem - problém splnitelnosti booelovských formulí
SBPSO	species-based PSO - PSO založené na třídách
SDS	stochastic diffusion search - stochastické rozptýlené prohledávání
SOMA	self-organizing migration algorithm - samoorganizující se migrační algoritmus
SPPSO	small population PSO - PSO s malou populací
SPSO	stretching PSO - rozpínavé PSO
TSP	traveling salesman problem - problém obchodního cestujícího
$\vec{v}_i$	vektor rychlosti částice
v_{max}	maximální rychlost částice
VEGA	vector evaluated genetic algorithm - vektorově ohodnocený genetický algoritmus
VEPSO	vector evaluated PSO - vektorově ohodnocené PSO
$\vec{x}_i$	vektor pozice částice
α	práh udávající pravděpodobnost s jakou bude částice přitahována k její nejlepší pozici
β	práh udávající pravděpodobnost přitahování k nejlepší částici
φ	akcelerační konstanta
φ_{ic}	konstanta setrvačnosti
χ	omezující faktor

Seznam příloh

Příloha A: Obsah přiloženého CD

Na CD, které je přiloženo k této práci, mají adresáře následující obsah:

Dokumenty

- Diplomová práce.pdf (Elektronická verze diplomové práce)
- Diplomová práce.doc (Zdrojový tvar diplomové práce)

Program

- Zdrojové soubory programu v jazyce Matlab (calculate.m, calculate2.m, clusteringPSO.m, gui.m, TSP_PSO.m)
- Datový soubor s lokacemi měst České republiky (CR.xls)