

Srovnání klasických a moderních optimalizačních metod

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

ABSTRAKT

Abstrakt česky

Optimalizační metody mají za úkol provádět různorodé činnosti a dojít k závěru pro konkrétní použití. Tyto metody nacházejí uplatnění ve všech oborech lidské činnosti. Úkolem této práce je vysvětlení a porovnání často používaných metod. V rámci klasických metod jsem nastínil základní princip operačního výzkumu a u metod moderních jsem se zaměřil na dva algoritmy s nejširším rozsahem použití – algoritmus SOMA a diferenciální evoluci.

Abstrakt ve světovém jazyce

Objective of optimization methods is to do many different actions and to find the solve for concrete applications. Optimization methods are useable in all domain human activities. Objective of this work is to explain and compare often using methods. I described basic principles of operation research in classic methods chapter and I specialized on two modern method algorithms with the widest size of use - the SOMA algorithm and differential evolution.

V rámci této práce bych chtěl poděkovat panu RNDr. Ing. Miloši Krčmářovi, za všechny rady a připomínky, který mi velmi pomohly při sestavování této práce.

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně, 02. 06. 2005

jméno diplomanta

OBSAH

OBSAH	5
ÚVOD.....	7
1 ZÁKLADNÍ POJMY.....	9
2 KLASICKÉ A MODERNÍ OPTIMALIZAČNÍ METODY	10
2.1 KLASICKÉ OPTIMALIZAČNÍ METODY	11
2.2 MODERNÍ OPTIMALIZAČNÍ METODY	12
3 DEFINICE ÚČELOVÉ FUNKCE	14
3.1 ÚČELOVÁ FUNKCE.....	14
3.2 ÚČELOVÁ FUNKCE JAKO GEOMETRICKÝ PROBLÉM.....	15
3.3 TVORBA ÚČELOVÉ FUNKCE.....	15
4 OPERAČNÍ VÝZKUM A JEHO VÝZNAM	16
4.1 HISTORIE	16
4.2 MATEMATICKÉ MODELOVÁNÍ.....	17
4.3 KLASIFIKACE DISCIPLÍN OPERAČNÍHO VÝZKUMU	20
5 LINEÁRNÍ PROGRAMOVÁNÍ.....	25
5.1 FORMULACE ÚLOH LINEÁRNÍHO PROGRAMOVÁNÍ.....	25
5.2 PŘÍKLADY ÚLOH LINEÁRNÍHO PROGRAMOVÁNÍ	26
6 SOMA ALGORITMUS	28
6.1 DEFINIČNÍ OBOR ALGORITMU SOMA	28
6.2 PARAMETRY SOMA ALGORITMU	29
6.3 POPULACE	31
6.4 MUTACE.....	31
6.5 KŘÍŽENÍ.....	32
6.6 PRINCIP ALGORITMU SOMA	32
6.7 STRATEGIE SOMA ALGORITMU	34
6.8 ZÁVISLOST SOMA ALGORITMU NA ŘÍDÍCÍCH A UKONČOVACÍCH PARAMETRECH	35
6.9 OBECNÉ PŘÍKLADY NA SOMA ALGORITMUS	36
7 DIFERENCIÁLNÍ EVOLUCE	37
7.1 PARAMETRY A TERMINOLOGIE	37
7.2 POPULACE	38
7.3 MUTACE.....	38
7.4 KŘÍŽENÍ.....	38

7.5	PRINCIP ČINNOSTI DIFERENCIÁLNÍ EVOLUCE	39
7.6	VARIANTY DIFERENCIÁLNÍ EVOLUCE	42
7.7	STAGNACE.....	43
7.7.1	Vliv mutace F	44
7.7.2	Vliv křížící konstanty CR.....	44
7.7.3	Vliv velikosti populace NP	45
7.7.4	Stagnace- doporučená nastavení parametrů	45
7.8	OBECNÉ PŘÍKLADY NA DIFERENCIÁLNÍ EVOLUCI	45
8	SROVNÁNÍ KLASICKÝCH A MODERNÍCH OPTIMALIZAČNÍCH METOD	46
8.1	ÚČELOVÁ FUNKCE	46
8.1.1	Operační výzkum	46
8.1.2	Evoluční algoritmy	46
8.2	PODMÍNKY OMEZENÍ	46
8.2.1	Operační výzkum	47
8.2.2	Evoluční algoritmy	47
8.3	ZÁKLADNÍ STANOVENÍ PODMÍNEK	47
8.3.1	Operační výzkum	47
8.3.2	Evoluční algoritmy	48
8.4	SHRNUTÍ.....	48
	ZÁVĚR	49
	SEZNAM POUŽITÉ LITERATURY	50
	SEZNAM OBRÁZKŮ	51
	SEZNAM TABULEK.....	52
	SEZNAM PŘÍLOH.....	53

ÚVOD

Už několik desetiletí je zřejmé úsilí člověka vykonávat mnoho činností tak, aby z určitého hlediska vykazovaly nejlepší možný výsledek. Jde o rozmanité lidské a ekonomické činnosti, výrobní a technologické procesy, experimenty apod. Hlediska můžou být ekonomická, výrobní, ekologické apod. Mnoho takových úloh je možné matematicky formulovat ve formě optimalizačních problémů nebo úloh optimalizačního řízení. Optimalizační algoritmy jsou velmi důležité při řešení mnoha problémů jak v inženýrské praxi, ekonomické činnosti nebo třeba i ve vojenské technice. Většinou se používají tam, kde je řešení daného problému analytickou cestou silně nevhodné či nereálné.

Pojem optimalizace je svým způsobem starý jako samo lidstvo. Úkolem člověka bylo vždy něco zlepšovat tím způsobem, aby dosáhl nejlepších výsledků a stálo ho to co nejméně úsilí a práce. Některé typy úloh již byly vymyšleny v období antiky. Již tenkrát se hovořilo o nalezení největších a nejmenších veličin. Jedná se o úlohy, které slouží k řešení nějakého pevně daného problému a které mají pouze jedno možné řešení. V případě, že budeme řešit problém v rámci ekonomického hlediska, firma se bude vždycky snažit vyřešit maximalizaci zisku, minimalizaci nákladu či třeba zvolení nejkratší cesty mezi dodavatelem a odběratelem. I tento fakt může někdy způsobit kritickou chybu při rozhodování. V případě řešení problému v inženýrské praxi může být definován optimalizační problém jako např. nalézt optimální trajektorii robota, nalézt optimální parametry regulátoru nebo například optimální přiřazení balistických raket na vojenské cíle ve vojenské technice. Pro problémy složitějšího typu byla vymyšlena asi před čtvrt stoletím skupina algoritmů, které byly schopny řešit velmi složité problémy efektivním způsobem.

Úkolem této bakalářské práce je shrnout nejčastěji používané metody z oblasti optimalizace a provést srovnání mezi nimi. Mým cílem bylo uvést čtenáře do problematiky těchto metod a vysvětlit několik základních principů jejich používání.

Z důvodu, kdyby četl práci někdo neznalý, jsem se v první kapitole snažil vysvětlit několik základních pojmů z oblasti optimalizace. Mým úkolem zde bylo vystihnout ty pojmy, které jsem dále v této práci používal a které jsou nezbytné pro pochopení další činnosti v oblasti optimalizace.

V druhé kapitole jsem nastínil přehled problematiky, kterou budu zpracovávat. Cílem bylo poskytnout všeobecný přehled o klasických a moderních metodách, dále také o

rozdělení evolučních algoritmů do tříd, které podávají jeden z možných pohledů na probírané metody. V této práci jsem se zaměřil pouze na tři typy metod, z nichž jedna spadá do oblasti klasických a zbývající dvě do oblasti moderních metod. Dále jsem se zmínil ještě o několika metodách, které také spadají do dané oblasti a jejich význam je neméně důležitý.

U všech optimalizačních metod vždy vycházíme z tzv. účelové funkce, a proto jsem podle [1] vysvětlil její význam. Tato kapitola má za úkol vysvětlit definici účelové funkce a co znamená lokální a globální extrém. Tyto principy se týkají celé bakalářské práce a jsou nezbytné pro pochopení následujících kapitol, kde jsou probrány podrobněji.

Ve třech dalších kapitolách jsem se věnoval důkladnému prozkoumání tří metod, které hrají v rámci klasických a moderních metod významnou roli. Jako první jsem se věnoval operačnímu výzkumu. Zde jsem se zaměřil na obecný postup při určování výrobního programu (např. co je důležité zvolit a z jakého důvodu) a dále jsem také zjednodušeně vysvětlil princip některých dalších disciplín, jako například teorii grafů, bez kterých by operační výzkum ztrácel na významu.

Jako jednu z nejzákladnějších disciplín jsem v následující kapitole uvedl lineární programování, které přímo navazuje na matematické programování a v podstatě je jeho součástí. Zde jsem také uvedl teoretické příklady, které jsou důležité pro pochopení stanovení podmínek při plánování výrobního programu a stanovení cílů.

V následujících dvou kapitolách jsem se věnoval dvěma zajímavým algoritmům, které fungují na podobném principu. Jedná se o algoritmus SOMA a diferenciální evoluci. Vždy vycházejí z populace a jedinců, kde každý má argumenty účelové funkce a každý z nich nese v podstatě řešení problému.

Posledních dvě kapitoly jsem věnoval srovnání klasických a moderních metod a zhodnocení dosažených výsledků.

1 ZÁKLADNÍ POJMY

- **Optimalizace** - proces interaktivního zlepšování řešení vzhledem k požadavkům, jaké má řešení splňovat. Cílem je najít nejlepší řešení při co nejmenším počtu kroků.
- **Operační výzkum** – vědní disciplína, která je zaměřena na analýzu různých typů rozhodovacích problémů
- **Model** - jedná se o systém předpokladů, rovnic, obrazů nebo vztahů atd., které velmi jednoduše vystihují některé aspekty fenomenálního prostředí. Modely mohou být vytvářeny různými způsoby. Můžou být založeny na teorii nebo hypotéze, nebo spolu mohou vzájemně souviset atd. V našem případě jsou většinou vytvářeny v procesu, který simuluje biologickou evoluci (evoluční algoritmy, genetické algoritmy) nebo pokud je nějaký systém analyzován pomocí operačního výzkumu.
- **Algoritmus** - přesně popsáný postup výpočtu, resp. obecně hledání řešení úlohy.
- **Evoluce** - biologický vývoj v nejobecnějším smyslu, proces zdokonalování živé hmoty a forem života od jednoduchých organismů ke stále složitějším, od nižších forem k vyšším formám, vrcholící vývojem člověka jako živočišného druhu
- **Evoluční algoritmy** – jedná se o společné označení pro optimalizační a aproximační algoritmy, které využívají evoluční mechanismy. Cílem těchto mechanismů je nalézt optimální konfigurace pro daný systém, na který jsou kladeny speciální požadavky, které se dosahují postupným zlepšováním. Smyslem evolučních algoritmů je cyklické vytváření nových populací a náhrada populací starých pomocí právě vytvořených nových populací.
- **Genetické algoritmy** – jedná se o vyhledávací algoritmy v rámci počítačového učení, které jsou součástí evolučních algoritmů. Jejich princip spočívá ve vytváření nových kandidátů na řešení problému, kteří vycházejí z nejúspěšnějších kandidátů (většinou z dvou nejúspěšnějších - rodičovských) předchozího kola hledaného řešení. Na rozdíl od evolučních výpočtů, ve kterých probíhá hledání řešení nad populací, která vzniká z rodičovské populace mutací nebo křížením.

2 KLASICKÉ A MODERNÍ OPTIMALIZAČNÍ METODY

V této kapitole se budu věnovat rozdělení optimalizačních metod na metody, které se vyvíjejí již poměrně dlouhou dobu a metody, které byly zformulovány nedávno. Budu se zabývat jak metodami uplatňující se v ekonomickém rozhodování (operační výzkum), tak i metodami, jejichž význam spočívá v nalezení různorodých problémů.

Optimalizační metody	
Klasické	Moderní
Operační výzkum	Soma Algoritmus
Horolezecký algoritmus	Diferenciální evoluce
Evoluční strategie	Mravenčí kolonie
Zakázané prohledávání	Rojení částic
	Rozptylové prohledávání
	Memetické algoritmy
	Metody imunologického systému

tabulka 1 – Porovnání klasických a moderních optimalizačních metod

Mezi klasické a moderní metody samozřejmě patří mnoho dalších metod, kterými se zabývat by bylo nad rozsah této práce. Zde jsem naznačil ty, které jsou podle mě nejzajímavější a samozřejmě nejdůležitější z oblasti optimalizace.

Různé algoritmy uvedené v tabulce patří do jednotlivých tříd algoritmů. Tyto třídy jsou různě efektivní a složité. Rozdělení tříd není jediné možné, vzhledem k tomu, že však vystihuje současný stav, lze je brát jako jeden z možných pohledů na klasické a moderní optimalizační metody. Jedná se o tyto třídy algoritmů:

- **Enumerativní** – výpočet všech možných kombinací určitého problému. Tyto algoritmy se používají v případě, když argumenty účelové funkce jsou diskrétního charakteru a nabývají malého množství hodnot.
- **Deterministické** – jedná se o algoritmy postavené na metodách klasické matematiky, které často vyžadují předběžné předpoklady umožňující této metodě podávat efektivní výsledky. Mezi dané předpoklady většinou patří konvexnost či lineárnost problému, malý prohledávaný prostor možných řešení, účelová funkce má pouze jeden extrém atd. Výsledkem těchto algoritmů je pouze jedno řešení. Mezi tyto algoritmy patří např. Horolezecký algoritmus atd.

- **Stochastické** – tento typ algoritmů je založen na využití náhody. Mezi tyto algoritmy patří například stochastický horolezecký algoritmus, zakázané prohledávání a evoluční strategie. Dané algoritmy jsou pomalé a vhodné jen pro prohledávané prostory možných řešení, které mají malý rozsah argumentů účelové funkce. K těmto algoritmům se zmíním v rámci klasických optimalizačních metod.
- **Smíšené** – jedná se o tzv. „mix“ metod deterministických a stochastických, které ve spolupráci dosahují dobrých výsledků. Do těchto algoritmů spadají i evoluční algoritmy. Tento typ algoritmů patří mezi výkonné a efektivní z důvodu, že jsou schopny nalézat kvalitní řešení během relativně malého počtu ohodnocení účelové funkce a také robustní, což znamená, že nejsou závislé na počátečních podmínkách a velmi často naleznou požadované kvalitní řešení během relativně malého počtu ohodnocení účelové funkce. Jsou dále také schopny nalézt více řešení během jednoho spuštění. Mezi tento typ algoritmů patří algoritmy, kterým se budu v této práci věnovat nejvíc, a to: SOMA algoritmus, diferenciální evoluce a dále také mravenčí kolonie, rojení částic, rozptylové prohledávání či mimetické algoritmy .

2.1 Klasické optimalizační metody

Mezi klasické optimalizační metody patří **operační výzkum**, jehož metodami se budu zabývat dále v následujících kapitolách.

Dále mezi tento typ metod patří algoritmy, které spadají do **stochastických algoritmů**, tj. ten typ algoritmů, které jsou založeny na využití náhody. Tyto algoritmy jsou čistě náhodné hledání hodnot argumentů účelové funkce a výsledkem je vždy nejlepší řešení, které bylo nalezeno v průběhu náhodného hledání. Tyto algoritmy jsou pomalé, vhodné jen pro prohledávané prostory možných řešení, které jsou malé a vhodné pro hrubý odhad. Mezi tento typ algoritmů patří např.:

1. **Stochastický horolezecký algoritmus.** Tento algoritmus prohledává prostor možných řešení ve směru největšího spádu. Velmi často uvázne v lokálním extrému díky své gradientní povaze. Základní princip spočívá v tom, že vždy vychází z náhodného bodu v prostoru možných řešení. Pro navrhnuté řešení se pomocí konečného souboru transformací navrhne dané okolí a daná funkce se minimalizuje jen v daném okolí. Zjištěné lokální řešení se použije jako střed pro výpočet nového okolí.

2. **Evoluční strategie.** Jeden z nejúspěšnějších stochastických algoritmů v historii. Vznikl na počátku šedesátých let a na rozdíl od jiných stochastických algoritmů pracuje přímo s reálnými hodnotami. Evoluční strategie spočívá v řešení ve formě vektoru x , které je mutováno pomocí vektoru náhodných čísel. Více o tomto algoritmu se dočtete v [3].
3. **Zakázané prohledávání.** Tento algoritmus spočívá ve vylepšení horolezeckého algoritmu, a to tak, že je do něho zavedena tzv. krátkodobá paměť, jejímž úkolem je pamatovat si transformace, pomocí nichž byl vypočítán aktuální střed. To způsobuje, že v konečném stavu nedochází k zacyklení díky zakázanému použití těchto transformací. Druhá varianta spočívá ve vylepšení o tzv. dlouhodobou paměť, která obsahuje transformace, které se nenacházejí v paměti krátkodobé, ale častokrát se používaly. Tento algoritmus na rozdíl od horolezeckého snadno neuvízne v lokálních extrémech.

2.2 Moderní optimalizační metody

Mezi moderní optimalizační metody patří algoritmy řešící velmi komplikované problémy efektivním způsobem. Daná třída algoritmů nese název „evoluční algoritmy“. Mezi typické rysy daných algoritmů patří to, že pracují s tzv. populacemi možných řešení, které se nazývají jedinci. Ti ovlivňují svou kvalitu na základě určitých evolučních principů v cyklech, které se nazývají „Generace“. Hlavním úkolem daného evolučního procesu je nalézt nejlepší možné řešení.

Dva hlavní algoritmy, kterými se budu v rámci moderních optimalizačních metod zabývat, nesou název **SOMA** a **diferenciální evoluce**. Pro jejich důležitost jim věnuji samostatné kapitoly. Z důvodu, že o každém z dalších moderních algoritmů by se dala napsat celá publikace se zaměřím pouze na pár algoritmů, které popíši velmi stručnou charakteristikou.

1. **Optimalizace mravenčích kolonií.** Činnost tohoto algoritmu vychází z chování mravenců v kolonii. Princip fungování je založen na tom, že existuje zdroj mravenců (mraveniště) a cíl jejich snažení (potrava). Po určité době vypuštění mravenců z mraveniště dochází k tomu, že se všichni mravenci pohybují po kratší cestě mezi zdrojem a cílem. To je způsobeno značkováním cesty feromonem. Intenzita feromonu ovlivňuje rozhodnutí mravence. V případě, že dorazí k rozcestí dvou cest vedoucích ke

stejnému cíli první mravenci, jejich rozhodování ke kterému cíli se vydají je náhodné. Mravenci, kteří zvolí kratší cestu ji označují a při návratu jsou díky značkám ovlivněni ve prospěch kratší cesty. Při návratu je cesta označena podruhé, což zvyšuje pravděpodobnost rozhodování dalších mravenců v její prospěch. Tento princip je uskutečněn v daném algoritmu.

2. **Rojení částic.** Algoritmus je založen na práci s populací jedinců, jejichž pozice v prostoru možných řešení je měněna pomocí tzv. rychlostního vektoru. V základní verzi tohoto algoritmu nedochází mezi jedinci k vzájemnému ovlivňování. Pokud používáme verzi s tzv. sousedstvím, dochází k odstranění vzájemného ovlivňování. Zde se definuje v populaci sousedství jedince „i“, které zahrnuje sousední jedince v rozsahu „ $i \pm$ velikost sousedství“. Z toho vyplývá, že v rámci tohoto sousedství pak dochází k ovlivňování mezi jedinci, kteří patří do jednoho sousedství a putují k nejhlubšímu extrému, který byl nalezen v tomto sousedství.
3. **Rozptylové prohledávání.** Vektorově orientovaný algoritmus, který generuje nové vektory (řešení) na základě pomocných heuristických technik. Při startu se vychází z řešení, která získáme na základě vhodné heuristické techniky. Dále jsou generována nová řešení na základě podmnožiny nejlepších možných řešení ze startu. Z daných možných řešení se opět vybere množina těch nejlepších a celý proces se opakuje.
4. **Memetické algoritmy.** Řešení těchto algoritmů spočívá v použití různých aproximačních algoritmů, technik lokálního vyhledávání apod. Tyto algoritmy mohou být také definovány jako strategie soutěže a spolupráce projevující atributy synergetiky.
5. **Metoda imunologického systému.** Tento algoritmus je založen na principech fungování imunologického systému v živých organismech. V tomto systému se nalézají různé typy agentů, které mají svůj specifický úkol. Dále také mohou komunikovat s jinými agenty. V rámci této komunikace vzniká hierarchická struktura schopná řešit komplikované problémy.

3 DEFINICE ÚČELOVÉ FUNKCE

Vzhledem k tomu, že v každé optimalizační metodě se snažíme dosáhnout nějakého určitého cíle optimalizace (maxima či minima), tzv. účelové funkce, je nutné si vlastně přesně matematicky vysvětlit, co účelová funkce znamená.

3.1 Účelová funkce

Účelovou funkcí rozumíme funkci, jejíž optimalizace (nalezení minima a maxima) povede k nalezení optimálních hodnot jejich argumentů. Podle [1] má funkce v bodě x_0 lokální maximum, pokud existuje okolí bodu x_0 takové, že platí:

$$f(x) \leq f(x_0) \quad (1)$$

pro všechna x z tohoto okolí. Funkce má v bodě x_0 ostré lokální maximum, pokud existuje okolí bodu x_0 takové, že

$$f(x) < f(x_0) \quad (2)$$

pro všechna x z tohoto okolí, s výjimkou bodu $x=x_0$. Pokud je X nějaká neprázdná množina z euklidovského N -rozměrného prostoru E_n , potom funkce f má v bodě $x_0 \in X$ lokální maximum vzhledem k X , pokud funkce f na množině X definována a pokud existuje okolí bodu x_0 takové, že (1) platí pro všechna x z tohoto okolí, která jsou současně body X . Ostré lokální maximum vzhledem k X se zavádí zcela obdobně, pouze místo (1) se vyžaduje platnost (2) při $x \neq x_0$.

Funkce f má v $x_0 \in X$ globální maximum vzhledem k X , pokud (1) platí pro všechna $x \in X$. Funkce f má v bodě $x \in X$ ostré globální maximum vzhledem k X , pokud (2) platí pro všechna $x \in X$, až na výjimku $x=x_0$. Pokud má f v bodě x_0 maximum, nazývá se x_0 bodem maxima funkce f (maximum ostré, lokální apod.). Z toho vyplývá, že funkční hodnotu $f(x_0)$ nazýváme maximální hodnotu funkce.

Pokud chceme definovat minimum funkce, provedeme tak že ve vztazích (1) a (2) se obrátí smysl nerovnosti. Výrazem extrém se tedy myslí buď maximum, nebo minimum dané funkce. Má-li mít funkce f v bodě x lokální extrém, musí být definována v nějakém okolí tohoto bodu x . Na druhé straně funkce f může mít v $x \in X$ extrém vzhledem k množině X , aniž je v nějakém okolí bodu x definována. Pokud f má extrém vzhledem

k X musí být vždy na X definována. U globálních extrému to vyplývá z toho, pokud se požaduje $f(x) <$ nebo $=$ nebo $> f(x_0)$, musí především oba symboly $f(x)$ a $f(x_0)$ mít smysl.

3.2 Účelová funkce jako geometrický problém

Každou účelovou funkcí můžeme rozumět geometrický problém, v jehož rámci se hledá nejnižší (minimum) či nejvyšší (maximum) pozice na N rozměrné ploše (hyperplocha, prostor možných řešení) daného problému. Počtem dimenzí N rozumíme počet optimalizovaných argumentů účelové funkce.

3.3 Tvorba účelové funkce

V rámci optimalizačního procesu se jedná o jeden z nejkritičtějších kroků. Jeho správné provedení výrazně ovlivňuje kvalitu výsledků. Z důvodu, že množina problémů, které lze nadefinovat jako optimalizační, je velmi rozsáhlá, neexistuje přesný návod konkrétních postupů jak účelovou funkci sestavit. Můžeme pouze naznačit obecné postupy a principy. Při řešení optimalizačního problému je nutné vždy vyřešit, čeho se má dosáhnout a z čeho můžeme vycházet.

Při tvorbě účelové funkce vycházíme z toho, že pokaždé hledáme extrémy typu minimum či maximum. Tohle hledání můžeme bez problémů převést na hledání jednoho typu extrému. Tím většinou bývá nalezení minima. Převedení se provede tak, že se účelová funkce vynásobí hodnotou -1 . To způsobí převrácení v opačnou funkci při nezměněných souřadnicích – hodnotách argumentů účelové funkce pro daný extrém.

4 OPERAČNÍ VÝZKUM A JEHO VÝZNAM

Jak jsem již naznačil, operační výzkum patří mezi klasické optimalizační metody. Jedná se o vědní disciplínu, která se zabývá zkoumáním operací v rámci nějakého systému. Je možné ho charakterizovat jako vědní disciplínu nebo spíše soubor relativně samostatných disciplín, které jsou zaměřeny na analýzu různých rozhodovacích problémů. Cílem je přitom stanovit takovou úroveň uskutečnění daných operací nebo jejich vzájemný vztah, aby bylo zajištěno co možná nejlepší fungování celého systému. Pro posouzení činnosti daného systému je třeba stanovit určitá kritéria. Provádění operací v systému nemůže být absolutně nezávislé – ve všech případech závisí na omezených zdrojích, které jsou při těchto operacích čerpány, na provádění jiných operací, na vnějších činitelích ovlivňujících chod systému apod. Operační výzkum je možné tedy charakterizovat i jako prostředek pro nalezení nejlepšího (optimálního) řešení daného problému při respektování celé řady různorodých omezení, které mají na chod systému vliv. Operační výzkum nachází aplikace všude tam, kde se jedná o analýzu a koordinaci provádění operací v rámci nějakého systému.

Řada metod operačního výzkumu doznala široké uplatnění v praxi. Některé z těchto metod mají postavení samostatných disciplín jako např. lineární programování. Z dalších disciplín nebo metod, jež se uplatňují v operačním výzkumu, uveďme např. teorii front, teorii zásob, teorii her či metody síťové analýzy.

4.1 Historie

Není jednoduché a v podstatě ani možné datovat vznik operačního výzkumu jako samostatné disciplíny. Počátky operačního výzkumu spadají už do 30. a 40. let minulého století a jsou spjaty mimo jiné s takovými jmény jako je G. B. Dantzig nebo nositel Nobelovy ceny za ekonomii L. Kantorovič. Rozvoj této disciplíny nastává jednak během 2. světové války, kdy byly vytvořeny ve Velké Británii a USA speciální týmy pracovníků pro analýzu složitých strategických a taktických vojenských problémů a operací, ale především potom během 50. let, ve kterých dochází ve světě k bouřlivému poválečnému ekonomickému rozvoji. Rozvoj operačního výzkumu a jeho jednotlivých disciplín vyplýval v tomto období skutečně z praktických potřeb – například některé metody a postupy dále obecně používané byly vyvinuty v rámci konkrétních praktických studií.

Dalším faktorem, ovlivňujícím rozvoj operačního výzkumu, je rozvoj výpočetní techniky. O rozvoji a významu operačního výzkumu svědčí i intenzita, s jakou jsou pořádány odborné konference, i to, jak vycházejí nové publikace a jsou pravidelně vydávány odborné časopisy speciálně s tímto zaměřením.

4.2 Matematické modelování

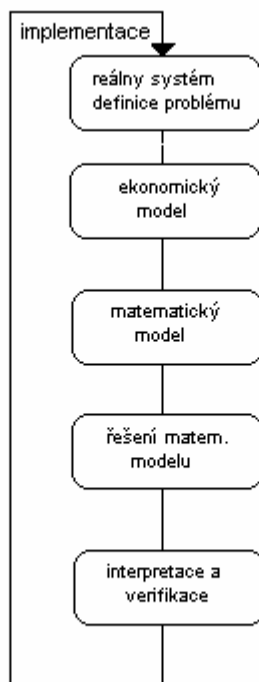
Základní nástrojem operačního výzkumu je *matematické modelování*. Pokud je tedy analyzován nějaký systém pomocí operačního výzkumu, potom tato analýza využívá model tohoto systému. Při analýze reálného systému prostřednictvím jeho modelu je třeba brát v úvahu, že model je pouze zjednodušeným obrazem tohoto systému. Modelování však má celou řadu výhod, pro které se stává často jediným prostředkem pro studium modelovaného systému. Jeho základní výhody modelového přístupu lze uvést:

- použití matematických modelů umožňuje strukturalizaci systému a specifikaci všech přípustných variant stavu systému, kterých může být často neomezené množství,
- modely umožňují analýzu chování systému ve zkráceném čase – procesy, které mohou trvat v reálném systému dny, měsíce či roky, mohou být simulovány na počítačích ve zlomcích sekund,
- s modely lze snadno manipulovat a provádět četné experimenty pomocí změn jejich parametrů,
- náklady na realizaci modelu nejsou sice zanedbatelné, jsou však vždy nižší než při experimentování s reálným systémem.

Při aplikaci kteréhokoli z odvětví operačního výzkumu pro řešení reálného rozhodovacího problému lze rozlišit několik základních, na sebe navazujících fází. (viz. Obr. 1)

1. **Rozpoznání problému v rámci reálného systému** a jeho definice je prvním podstatným krokem aplikace modelů operačního výzkumu. Zde je třeba vyzvednout roli vedoucích pracovníků na různých úrovních, kteří by měli být schopni rozpoznat problém, odhadnout potřebu modelového přístupu pro jeho analýzu a případně vytvořit tým příslušných odborníků, který se na ní bude podílet.

2. **Formulace ekonomického modelu** daného problému. Reálný systém je zpravidla příliš složitý na to, aby bylo možné modelově postihnout všechny jeho stránky. Při studiu konkrétního problému se však ukazuje, že není nutné a třeba ani žádoucí všechny prvky systému uvažovat.



Obr. 1 - Fáze při aplikaci operačního výzkumu

Ekonomický model lze charakterizovat jako zjednodušený popis reálného systému, který obsahuje s ohledem na analyzovaný problém pouze ty nejdůležitější prvky a vazby mezi nimi. Ekonomický model by měl obsahovat především:

- *Cíl analýzy*, tzn. pokud možno jednoznačné určení cílového stavu modelovaného systému – cílem přitom může být například maximalizace zisku při plánování výrobního programu firmy, minimalizace rizika při řízení investiční strategie apod.
- *Popis procesů*, které v systému probíhají – pod pojmem proces zde budeme rozumět reálnou aktivitu, která v systému probíhá s nějakou intenzitou, která má vliv na cíl analýzy. Při plánování výrobního programu může být procesem například výroba výrobku a intenzitou procesu objem výroby tohoto výrobku – objem výroby má potom vliv na cíl analýzy, kterým může být maximalizace zisku, minimalizace nákladů apod.

- *Popis činitelů ovlivňujících provádění procesů* – v reálných úlohách nemohou být prováděny procesy s neomezenou intenzitou, naopak jejich realizace je ovlivňována celou řadou činitelů, které je třeba respektovat. Činiteli mohou být spotřeba omezených zdrojů při výrobě výrobků, potřeba dodržovat při výrobě zadané relace mezi výrobky apod.
 - *Popis vzájemného vztahu mezi procesy, činiteli a cílem analýzy* – například pokud je procesem výroba výrobku, činitelem spotřeba suroviny a cílem maximalizace zisku, potom je třeba určit, kolik suroviny se spotřebuje při výrobě výrobku a jaký zisk výroba tohoto výrobku přinese.
3. **Formulace matematického modelu** daného problému. Ekonomický model je v podstatě jakýmsi slovním a numerickým popisem problému. Aby bylo možné daný problém řešit, je třeba jej nějakým způsobem formalizovat – převést tedy ekonomický model na model matematický, který je potom řešitelný standardními postupy. Matematický model obsahuje stejné prvky jako model ekonomický, samozřejmě však v jiném vyjádření:
- *Cíl analýzy*, pokud je vůbec definován, je zpravidla vyjádřen ve formě lineární či nelineární funkce n proměnných.
 - *Procesům*, probíhajícím v systému, odpovídají v matematickém modelu proměnné, intenzity provádění procesů jsou potom vyjádřeny jako hodnoty těchto proměnných.
 - *Činitelé* v ekonomickém modelu mohou být v matematickém modelu vyjádřeny velmi rozličnými způsoby v závislosti na povaze analyzovaného problému – jedním z běžných způsobů je jejich vyjádření ve formě lineárních či nelineárních rovnic případně nerovnic.
 - *Vazby mezi procesy, činiteli a cílem analýzy* jsou popisovány pomocí neřiditelných parametrů (parametrů, jejichž hodnoty nemůže uživatel ovlivňovat)
4. **Vlastní řešení matematické modelu** je spíše technickou záležitostí. Lze pro něj použít metody a postupy, navržené v jednotlivých odvětvích operačního výzkumu. Role uživatelů se zde omezuje na výběr vhodného programového prostředku, použitelného pro řešení toho kterého typu modelu a na jeho formální obsluhu. Většina z metod operačního výzkumu je dnes totiž již zabezpečena poměrně kvalitními programovými systémy, bez kterých si nelze zpracování reálných úloh představit.
5. Důležitou otázkou aplikace modelů operačního výzkumu je **interpretace výsledků** získaných v předcházejícím kroku a jejich následná **verifikace**. Samotná interpretace

však nestačí, výsledky je třeba verifikovat a tím současně vlastně ověřit, zda byl ekonomický a následně matematický model problému sestaven správně. Pokud jsou při sestavování modelu opomenuty některé podstatné stránky systému, potom řešení modelu může být sice “optimální” v rámci tohoto modelu, ale v praxi se může ukázat jako nepoužitelné.

6. V případě úspěšné **verifikace** výsledků lze přistoupit k jejich implementaci v rámci analyzovaného reálného systému. Úspěšná implementace by měla potom samozřejmě přispět ke zlepšení fungování daného systému s ohledem na sledovaný a v modelu definovaný cíl.

4.3 Klasifikace disciplín operačního výzkumu

Modely operačního výzkumu se velmi odlišují a jejich úkolem je zabývat se rozdílnými oblastmi ekonomického života. Vzhledem k této skutečnosti vznikla potřeba specifických přístupů k řešení jednotlivých tříd problémů a postupem času vznikly relativně samostatné disciplíny či odvětví operačního výzkumu.

1. **Matematické programování** je odvětví operačního výzkumu. Toto odvětví se zabývá řešením optimalizačních úloh, které se zaměřují na nalezení extrému daného kritéria, definovaného ve tvaru kritériální funkce n proměnných, na množině variant určených soustavou omezujících podmínek, které zadáváme ve tvaru lineárních nebo nelineárních rovnic či nerovnic. Matematický model v úloze matematického programování lze popsat daným způsobem:

Maximalizovat nebo minimalizovat:

$$z = f(x_1, x_2, \dots, x_n) \quad (3)$$

za daných podmínek

$$g_1(x_1, x_2, \dots, x_n) \geq 0,$$

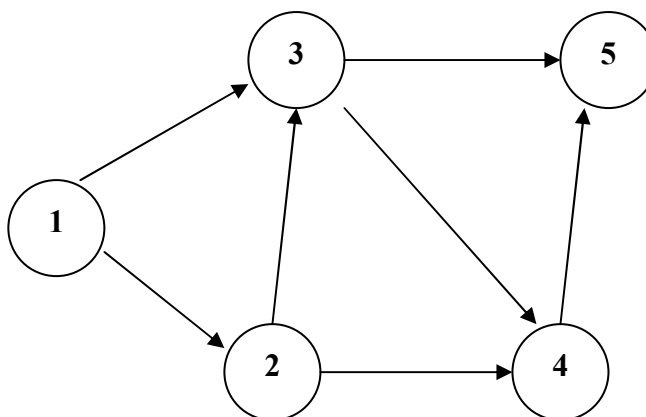
$$g_2(x_1, x_2, \dots, x_n) \geq 0,$$

$$g_m(x_1, x_2, \dots, x_n) \geq 0,$$

$$x_j \geq 0, j = 1, 2, \dots, n,$$

kde n je počet proměnných, m je počet jeho omezujících podmínek a $f(x)$, $g(x)$, $i=1,2,\dots,m$ jsou obecné funkce n proměnných. Z matematického hlediska se jedná o určení hodnot proměnných modelu x_j tak, aby bylo možné respektovat všechny omezující podmínky úlohy, a aby bylo možné dosáhnout daného extrému dané kritériální funkce. V případě, že kritériální funkce je lineární, a všechny rovnice a nerovnice použité v modelu jsou také lineární, potom se jedná o lineární programování. Pokud je alespoň jedna funkce nelineární, mluvíme o úloze nelineárního programování. Využívání úloh s použitím lineárního programování je velmi časté, naopak nelineární modely jsou výrazně méně používané z důvodu problému s jejich řešením. Mezi typicky aplikované oblasti lineárního programování patří např. určení strategie reklamy atd.

2. **Vícekritériální rozhodování** je podstatně mladší disciplína operačního výzkumu zabývající se analýzou rozhodovacích úloh, ve kterých jsou možnosti, které jsou k dispozici pro rozhodování ne podle jednoho, ale několika hodnotících kritérií zároveň. V klasickém případě nejsou hodnotící kritéria v souladu a cílem analýzy v úlohách vícekritériálního rozhodování je tedy vlastně řešení konfliktu mezi navzájem protikladnými kritérii. Úlohy vícekritériálního rozhodování se dělí na dvě skupiny, podle způsobu definování množiny rozhodovacích variant. Jde o tyto skupiny:
 - **vícekritériální hodnocení variant** – v případě, že varianty jsou určeny jejich konkrétním výpočtem či seznamem. Oblast aplikací úloh je velmi rozmanitá a ve většině případech souvisí s každodenním životem, jako např. přijímací řízení na vysokou školu (varianty: uchazeči, kteří se dostaví k přijímacímu řízení; kritéria: výsledky přijímacích zkoušek)
 - **vícekritériální programování** – v případě, že varianty jsou určeny soustavou podmínek stejných jako v úlohách matematického programování
3. **Teorie grafů** je velmi často v praxi používaným odvětvím operačního výzkumu. Grafy rozumíme objekty tvořené *uzly* a spojnice mezi nimi se nazývají *hrany*, pomocí kterých lze znázorňovat různé reálné systémy.



Obr. 2 – Ukázka jednoduchého grafu

Grafem může být tvořena například reálná komunikační síť s konkrétními místy (uzly) a spojnicemi mezi nimi (hrany). Na takové síti lze potom definovat řadu optimalizačních úloh, mezi které patří například hledání a nalezení nejkratší cesty mezi dvěma uzly v grafu. Hrany představují reálné činnosti tvořící projekt a posloupnost hran v grafu (určená šipkami) určuje návaznosti v provádění těchto činností. Každé činnosti (hraně) je přitom přiřazeno určité ohodnocení (doba trvání činnosti, náklady na činnost) a cílem rozboru je potom časový či nákladový rozbor realizace celého projektu. V reálných úlohách představují grafy reprezentaci nějakého reálného systému. To znamená, že každý prvek grafu (uzel i hrana) má nějakou interpretaci. Potom tedy může celý graf představovat určitou distribuční síť nějaké společnosti, kde uzly budou reálná místa (sklady, odběratelé a dodavatelé) a hrany budou mít podobu silničních či jiných spojení mezi těmito místy. Dále se také teorie grafů používá často pro optimalizaci řízení projektů (složitých činností) metody kritické cesty, PERT, GERT, jejichž úkolem je zkrátit dobu činnosti na co nejkratší.

4. **Modely řízení zásob** je odvětví operačního výzkumu, který se zabývá strategií řízení zásobovacího procesu a optimalizací objemu skladovaných zásob s ohledem na minimalizaci nákladů, případně ztrát, které souvisejí s udržováním, objednáváním a vydáváním zásob ze skladu. Základní charakteristikou v modelu řízení zásob je charakter poptávky po sledované jednotce zásoby. Jedná se buď o poptávku deterministickou, kde se jedná o poptávku v rámci uvažovaného časového období (například spotřeba polotovarů při výrobě nějakého výrobku je určena objemem výroby, který je pevně daný) nebo o poptávku stochastickou, kde můžeme její velikost

odhadnout pouze s určitou pravděpodobností (například poptávka po zboží nově uváděném na trh).

5. **Teorie hromadné obsluhy** se zabývá zkoumáním systémů, které obsahují dva základní typy jednotek – *požadavky*, které přicházejí do systému a potřebují obsluhu a *oblužné linky*, které tuto obsluhu realizují. S realizováním obsluhy souvisí také vytváření front. Systémy daného typu jsou běžné. Mezi takovými systémy může být obchod, zdravotnické zařízení, banka, výrobní linka apod. Cílem analýzy systému hromadné obsluhy je fungování celého systému tak, aby fungoval co nejefektivněji. Analýza je v tomto případě vlastně řešení konfliktu mezi stupněm využití obslužných linek a dobou čekání požadavků ve frontě na obsluhu. Se systémy hromadné obsluhy se můžeme setkat při každodenním životě, i když samozřejmě ne vždy se používají k jejich analýze nějaké modelové prostředky. Pokaždé máme nějaký systém (např. ordinace lékaře, banka), nějakou obslužnou linku (např. lékař, úředníci u přepážky) a samozřejmě nějaké požadavky (pacienti, klienti).
6. **Modely obnovy** mají za úkol prozkoumat systémy a nahradit jednotky systému, které selžou po určité době provozu a je třeba je opravit nebo nahradit novými. Doba bezporuchového provozu jednotek je přitom náhodná veličina. Cílem zkoumání je odhadnout věkovou strukturu jednotek a určit počty jednotek, které v jednotlivých časových obdobích selžou a bude je tedy třeba nahradit. Dále také musíme zajistit, aby náklady vzniklé při opravách strojů byly co nejmenší.
7. **Markovovy rozhodovací procesy** jsou obecný prostředek pro popis chování dynamických systémů. Jde o systémy, které se mohou ve sledovaných časových intervalech nacházet vždy v některém z konečného počtu stavů. Změna stavů systému v po sobě jdoucích obdobích podléhá náhodnému chování. Základním cílem Markovovy analýzy je určování budoucího chování daného systému. Pokud budeme sledovat chování systému v diskrétních časových okamžicích, pak se může systém nacházet v časovém okamžiku n v jednom z možných stavů s určitou pravděpodobností. V následujícím časovém okamžiku $(n+1)$ se může stav systému buď změnit nebo zůstane beze změny. Základní myšlenka Markovových procesů spočívá v tom, že výskyt stavu v čase $(n+1)$ vždy závisí pouze na výskytu stavu v čase předcházejícím (čas n). Mezi typické příklady těchto procesů patří například sledování chování výrobní linky na začátku každé hodiny. Z toho vyplývá, že linka může být ve

dvou stavech - pracuje nebo je ve stavu poruchy. Cílem je určit ve stanoveném časovém horizontu pravděpodobnost zda linka bude či nebude pracovat.

8. **Teorii her** lze popsat jako řadu rozhodovacích situací s více než jedním účastníkem (rozhodovatelem), které lze popsat jako hru, ve které mají jednotlivý hráči (rozhodovatelé) nějaké strategie svého chování na kterých závisí jejich výhra. Teorie her se zabývá průzkumem a definováním optimálních strategií v takovýchto rozhodovacích situacích.
9. **Simulace** je nejdůležitějším a často jediným možným nástrojem pro analýzu složitých systémů. V podstatě se nejedná o samostatnou disciplínu operačního výzkumu, jde spíše o prostředek analýzy, který je použitelný pro různé typy modelů. Simulace spočívá s experimentováním vytvořeného modelu daného systému na počítači. U ekonomických modelů probíhá počítačové zpracování ve srovnání s reálným systémem zpravidla ve zrychleném čase. Tímto způsobem je možné sledovat stav zkoumaného systému při změnách parametrů, které ovlivňují jeho chování a pokusit se tak tento systém optimalizovat. Pro simulaci nestačí jenom výkonné počítače, ale také potřebný software. U softwarových prostředků bývá zpravidla problémem to, že jsou velmi drahé. Přínos, který může jejich účelné využití přinést, však tuto cenu výrazně převyšuje.

5 LINEÁRNÍ PROGRAMOVÁNÍ

Lineární programování je disciplína operačního výzkumu, která se zabývá řešením rozhodovacích problémů, ve kterých jde o určení intenzit realizace procesů, které probíhají nebo mohou probíhat v daném systému. Je důležité respektovat všechny podmínky, které realizaci daných procesů ovlivňují, a najít nejlepší možné řešení, tak aby byl cíl rozhodování splněn co nejlépe. Termín lineární programování se skládá ze dvou slov, která poměrně přesně vystihují podstatu tohoto odvětví operačního výzkumu. Slovo programování není v tomto termínu chápáno jako programování v nějakém programovacím jazyku. S počítači má však společné to, že řešení úloh není možné bez použití výpočetní techniky. Programováním se zde spíše rozumí synonymum pro plánování nebo vytváření scénářů budoucího vývoje. Slovo lineární vyjadřuje, že všechny vazby v modelech lineárního programování jsou vazbami lineárními. Lineární programování je tedy prostředkem pro plánování uskutečnění určitých činností, které zabezpečují dosažení optimálního výsledku ve vztahu k definovanému cíli.

5.1 Formulace úloh lineárního programování

Zásadním krokem při aplikaci lineárního programování je formulování ekonomického a dále matematického modelu daného problému. Formulace ekonomického modelu je komplikovaný proces, který vychází ze zkoumání a poznání reálného ekonomického systému. Proto se v rámci bakalářské práce jím nebudu zabývat. Je možné rozpoznat základní směry aplikací a typické úlohy lineárního programování.

Jako první krok u úloh lineárního programování je definice a popis ekonomického modelu daného problému. U ekonomického modelu musíme definovat procesy, které v daném systému probíhají, definovat činitele, které omezují realizaci jednotlivých procesů, a určit cíle optimalizace. Nyní se budu zabývat jednotlivými kroky, které je důležité určit při postupu převedení ekonomické modelu na model matematický:

1. **Identifikace rozhodovacích proměnných (x_j).** Důležité je určení počtu rozhodovacích proměnných, stanovení jejich významu a fyzikálního rozměru. Proměnné jsou zpravidla definovány tak, aby odpovídaly jednotlivým procesům, které probíhají v daném systému. Návod na správné určení rozhodovacích proměnných může dát otázka “co je cílem analýzy, o čem máme rozhodnout?”.

2. **Definice optimalizačního kritéria** a vytvoření účelové funkce, která je funkcí rozhodovacích proměnných.
3. **Identifikace činitelů modelu** a jejich vyjádření ve formě omezujících podmínek. Pro správnost matematického modelu je důležité brát v úvahu všechny činitele a na žádné nezapomenout, jinak by mohl sestavený model být sice z formálního matematického pohledu správný, ale výsledky, který by tento model poskytoval, by mohly být nesmyslné a v reálném systému nepoužitelné. Součástí tohoto kroku je i určení vztahu mezi jednotlivými činiteli a procesy ve formě strukturních koeficientů modelu a určení pravých stran omezujících podmínek.

5.2 Příklady úloh lineárního programování

V následující části popíšu přehled typických úloh lineárního programování s uvedením možností, co můžou být *procesy*, *činitelé* a co může být *optimalizačním kritériem*. Přesný popis při řešení příkladů naleznete v [2]

1. Úlohy výrobního plánování (problém alokace zdrojů)

V těchto typech úloh se musí jako první určit sortiment výroby s tím, že je potřeba respektovat omezující podmínky na straně vstupů (omezená kapacita surovin, energie atd.) i na straně výstupů (odběratelská omezení určující minimální resp. maximální objem produkce). Cíl optimalizace může být maximalizace zisku nebo minimalizace nákladů a jako proměnné zvolíme objem produkce jednotlivých výrobků nebo třeba dobu produkce výrobku, po kterou se bude daný výrobek produkovat.

2. Úlohy finančního plánování (optimalizace portforlia)

Jedná se o standardní optimalizační úlohu. Cílem optimalizace je určit objem investic do jednotlivých investičních variant s cílem maximalizovat očekávaný výnos případně minimalizovat riziko. Proměnné budou představovat objemy investic. Omezující podmínky mohou odpovídat investiční strategii, která může stanovovat limity pro investice do jednotlivých druhů investičních variant.

3. Plánování reklamy

Toto plánování souvisí s rozvržením rozpočtu na reklamu do jednotlivých médií. Procesem je umístění reklamy do určitého média v konkrétním časovém okně dne v týdnu. Proměnnými jsou počty opakování reklamy v daném médiu. Omezující podmínky

vycházejí z omezeného rozpočtu, z definice cílové skupiny, na kterou se reklama zaměřuje. Mezi cíle optimalizace patří maximalizace mediálních charakteristik, která uvádí kolikrát se průměrně setká s reklamou jedinec z cílové skupiny

4. Úloha o dělení materiálu

Jedná se o problém dělení větších celků na menší části tak, aby byl minimalizován odpad. Je však nutné respektovat požadavky na poměr, v jakém mají vzniklé menší části být, kolik minimálně resp. maximálně jich má vzniknout apod. Tento typ úlohy se dělí na dvě části: jednorozměrnou a dvojrozměrnou. Jednorozměrná část znamená, že dělení je charakterizováno pouze jedním rozměrem. Tento problém odkazuje na úlohu lineárního programování. U dvojrozměrné části je důležité, že z plochy se vyřezávají menší díly. Je výrazně složitější než část jednorozměrná. U určování proměnných a procesů nastává menší problém. U proměnných vycházíme z toho, že každý větší celek lze na menší části rozřezat celou řadou způsobů. Procesem se zde rozumí použití jednoho z potenciálních způsobů. Procesům budou odpovídat proměnné a hodnoty proměnných budou udávat kolikrát se určitý způsob dělení použije.

5. Rozvrhování pracovníků

U úloh tohoto typu je důležité definovat řadu omezujících podmínek vycházejících například z kvalifikace pracovníků, z minimálního nebo pevně stanoveného počtu pracovníků, kteří musí být v dané okamžiku na směně atd. Procesem se rozumí přiřazení pracovníka v určitém čase na směnu, odpovídající proměnná může vyjadřovat, zda pracovník na dané směně bude nebo nebude přiřazen. (0 nebo 1)

6. Distribuční úlohy lineárního programování

Patří mezi ně například optimalizace distribuce zboží mezi dodavateli a odběrateli.

6 SOMA ALGORITHMUS

Činností tohoto algoritmu jsou geometrické principy. Mezi evoluční algoritmy patří z důvodů nevytvoření potomků během jeho běhu. Tento algoritmus by se dal spíše přirovnat k harémové tvorbě potomků ve stádu, než ke klasickému výběru rodičů z populace.

Původní myšlenkou tohoto algoritmu bylo napodobování chování skupiny inteligentních jedinců, kteří kooperují při řešení společného problému (např. hledání potravy).

Princip tohoto algoritmu vzniknul na základě principů, které lze odpozorovat v přírodě a kterými se v sociálně - biologickém prostředí řídí inteligentní jedinci, kteří spolupracují na řešení společného úkolu. Oproti jiným evolučním algoritmům zde neprobíhá tvorba nových řešení (jedinci, potomci) na základě metody křížení rodičů, ale je založena na kooperativním prohledávání (migraci) prostoru možných řešení daného problému. Vlastní jádro SOMA algoritmu nekopíruje zmíněné evoluční principy, ale řídí se principy vycházejícími ze spolupráce inteligentních jedinců migrujících v prostoru možných řešení tak jako jejich biologické protějšky (např. mravenci, včely) po krajině. Z toho důvodu byl pro tento evoluční cyklus známý jako „Generace“ zvolen název „Migrační kolo“.

6.1 Definiční obor algoritmu SOMA

Každý algoritmus provádí svou činnost na specifické množině problémů, která může mít pro každý algoritmus jiný rozsah. To souvisí s problémem, že nestačí mít jen dobrý algoritmus, ale také vědět s jakou množinou problémů je daný algoritmus schopný pracovat. Tuto množinu problémů nazýváme definičním oborem, na kterém se daná funkce definuje.

Optimalizačním problémem můžeme rozumět problém geometrický, u něhož je cílem nalézt nejnižší (minimum) či nejvyšší (maximum) bod na N rozměrné ploše. Algoritmus SOMA je velmi výkonný a vhodný pro globální optimalizaci. Z toho vychází definiční obor tohoto algoritmu, pro který musí platit atributy testovacích funkcí, na kterých byl algoritmus testován. Jedná se o tyto atributy, pro jejichž testovací funkce musí platit že:

- jsou nefraktálního typu (tzn. že jejich geometrický motiv se neopakuje v základním tělese až do nekonečna)

- jsou definované na reálných, celočíselných nebo diskrétních argumentech
- mají jeden či více extrémů
- mají různá omezení (v podobě kladených argumentů či hodnot účelové funkce)
- jsou víceúčelové
- jsou lineární, nelineární
- představují problémy typu „jehla v kupce sena“

6.2 Parametry SOMA algoritmu

Algoritmus je ovlivňován speciální množinou parametrů, které se dělí na dva druhy parametrů:

- **řídící** – parametry, které ovlivňují kvalitu běhu algoritmu (z hlediska hodnoty účelové funkce)
- **ukončovací** – parametry, které za předem nadefinovaných podmínek parametrů ukončují

Tyto parametry jsou zvoleny uživatelem ještě před začátkem běhu algoritmu. Doporučený rozsah parametrů pro SOMA algoritmus je zobrazen v tabulce:

Parametr	Doporučený rozsah	Poznámka
Mass	<1,1,5>	Řídící parametr
Step	<0,11, Mass>	Řídící parametr
PRT	<0,1>	Řídící parametr
D	dáno problémem	Počet argumentů účelové funkce
NP	<10, definuje uživatel>	Řídící parametr
Migrace	<10, definuje uživatel>	Ukončovací parametr
AcceptedError	< $\pm$ libovolný, definuje uživatel>	Ukončovací parametr

tabulka 2 – Parametry SOMA algoritmu

Je velmi důležité nastavit parametry tak, aby kvalita běhu algoritmu odpovídala nastavení těchto parametrů a zvolit konkrétní sub-intervaly parametrů. Zde je stručný popis jednotlivých parametrů a volení rozsahu tak, aby vykonávali dobře funkci při průběhu algoritmu.

- **Mass** – parametr určuje jak daleko se aktivní jedinec zastaví od tzv. vedoucího jedince. Při Mass = 1 se aktivní jedinec zastaví na pozici vedoucího jedince, při Mass=2 je za ním ve stejné vzdálenosti z jaké odstartoval. Pokud je Mass < 1, tak

se zastaví před vedoucím jedincem, a z toho vyplývá degenerace migračního procesu, z toho plyne, že budou nalezeny pouze lokální extrémy. Parametr je dobré mít nastaven jako $Mass > 1$. Konečná hranice, kdy $Mass=3$ byla zjištěna jako dostačující ve všech simulacích na všech problémech.

- **Step** – parametr určuje „zrnitost“, s jakou bude mapována cesta aktivního jedince. Pokud je účelová funkce unimodální (konvexní a nebo pár lokálních extrémů), je možné použít velkou hodnotu pro Step jako urychlení chodu algoritmu. Pokud neznáme přibližnou geometrii, které reprezentuje účelovou funkci, potom je dobré nastavit Step na nízkou hodnotu.
- **Perturbation** – tento parametr vytváří perturbační vektor (PRTVector), který ovlivňuje aktivního jedince, zda se bude pohybovat přímo k vedoucímu jedinci či ne. Tento parametr má největší citlivost s hodnotou okolo 0,1. Pokud narůstá hodnota tohoto parametru, pak konvergence SOMA k lokálním extrémům silně vzrůstá. Pokud je hodnota rovna jedné, potom stochastická složka chování SOMA zaniká a algoritmus se chová pouze podle deterministických pravidel (hodí se pouze k lokální optimalizaci).
- **Dimensions** – udává počet optimalizovaných proměnných daného problému. Proměnnými rozumíme počet argumentů účelové funkce. Tento parametr je možné změnit pouze tehdy, když chceme předefinovat celý problém.
- **Population** – parametr určuje kolik jedinců bude tvořit populaci. Minimální nastavení jedinců je už od 2, v tomto případě by tento algoritmus odpovídal klasickým deterministickým metodám. V případě jednoduché funkce můžeme nastavit nízký počet jedinců, v opačném případě je možné nastavit $NP=D$.
- **Migrations laps** – ekvivalent parametru Generace. Udává kolikrát se populace jedinců přeorganizuje.
- **AcceptedError** – parametr definuje, jaký je povolen maximální rozdíl mezi nejhorším a nejlepším jedincem v aktuální populaci. Pokud je rozdíl menší nežli AcceptedError, potom je běh algoritmu ukončen. Je dobré mít nastaven tento parametr na hodnotu jedna. V případě vysoké hodnoty se algoritmus zastaví dříve, nežli populace stačí lokalizovat globální extrém. Pokud je hodnota rovna nule, nebo v blízkosti nuly, dojde k zastavení algoritmu až po vyčerpání všech migračních kol.

6.3 Populace

Typickým rysem algoritmu SOMA, diferenciální evoluce a všech ostatních algoritmů je to, že jsou založeny na práci s populací jedinců. Populace je představena jako matice Population x Dimensions, kde sloupce představují jednotlivé jedince. Jednotlivý jedinec může představovat aktuální řešení daného problému. Jedná se o množinu argumentů účelové funkce, jejíž optimální číselná kombinace je hledána. Každý jedinec má v sobě hodnotu účelové funkce, která udává jak vhodný je jedinec pro další vývoj populace (viz. tabulka 6) Daná hodnota není součástí evolučního procesu, ale pouze nese informaci o kvalitě příslušného jedince.

Smyslem evolučních algoritmů je cyklické vytváření nových populací a náhrada populací starých pomocí právě vytvořené nové. Tento princip je nadefinován pomocí přesných matematických pravidel.

Před vytvořením populace je potřeba nadefinování tzv. vzoru, který nese označení Specimen. Jedná se o vzorového jedince, podle něhož se náhodně vygeneruje celá populace. Také obsahuje pro každý parametr jedince definici mezí a typ proměnné. Dále se používá pro korekci parametrů jedince při překročení hranic prohledávaného prostoru.

6.4 Mutace

Proces, při kterém se náhodně mění některé parametry daného jedince. U SOMA algoritmu mutace dostala název perturbace. Název perturbace vznikl z důvodu, že při pohybu jedinců skrz prostor možných řešení je jejich pohyb náhodou rušen (perturován) a nikoli mutován, což se hodí spíše k filozofii tvorby nových potomků pomocí páření, spadající do oblasti klasických genetických algoritmů. Jak silně bude tento proces probíhat (jak moc se změní parametry jedince) záleží na nastavení parametru **PRT**. Tento parametr generuje perturbační vektor, který se generuje zvlášť pro každého jedince a platí jen pro aktuální běh aktivního jedince.

Zvláštností oproti jiným algoritmům je to, že parametr perturbačního vektoru je tvořen ještě předtím, než jedinec začne putovat. Pokud se pohlíží na putování jedince jako na tvorbu nových potomků, pak je perturbační vektor vytvořen dříve než jsou noví potomci vytvořeni. U ostatních algoritmů je mutace dělana až po vytvoření nového potomka.

Výsledným efektem perturbačního vektoru je eliminace pohybu jedince z N rozměrného na N-k rozměrný prostor. To vede ke zvýšení šance nalézt globální extrém.

6.5 Křížení

Dochází k mapování prostoru, po kterém aktuální jedinec putuje po hyperploše po diskrétních skocích. Tím můžeme rozumět vygenerování sekvence potomků, z nichž přežije jen ten nejlepší. U SOMA algoritmu dochází ke generování sekvence potomků na rozdíl oproti genetickým algoritmům, kde se generují pouze dva potomci. Při putování jedince po hyperploše si každý jedinec pamatuje souřadnice pozice na níž našel nejlepší řešení v rámci své cesty a dané řešení pak postupuje do dalších migračních kol. Cestování jedince je dáno rovnicí:

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML})t \text{ PRTVector} \quad (4)$$

kde $t \in \langle 0, \text{po Step až po, Mass} \rangle$

kde vyplývá, že $x_{i,j}^{ML+1}$ je j-tý prvek i-tého jedince v migračním kole ML+1, $x_{i,j,start}^{ML}$ je j-tý prvek i-tého jedince v migračním kole ML, $x_{L,j}^{ML}$ je j-tý prvek vedoucího jedince (Leadera) v migračním kole ML. Z rovnice lze vyvodit, že PRTVector ovlivňuje pohyb jedince tak, že pokud má všechny prvky rovny 1, tak aktivní jedinec putuje rovnou podle směrového vektoru k vedoucímu jedinci (jedinec s nejlepší možnou hodnotou účelové funkce v daném migračním kole). Když se některé prvky rovnají 0, potom je druhý člen v obou rovnicích anulován a daná souřadnice se nemění. To způsobuje pohyb jedince v N-k rozměrném prostoru.

6.6 Princip algoritmu SOMA

Algoritmus SOMA vychází z chování inteligentních jedinců, kteří řeší společný problém. V tomto problému je společným úkolem to, že jedinci spolupracují(soutěží), i když to o sobě nevědí. Při spolupráci si jednotliví jedinci navzájem sdělují kvalitu řešení hledaného problému a snaží se podle toho přizpůsobit své chování. Ve fázi soutěžení se každý jedinec snaží vyhrát nad ostatními. Jakmile je ukončena fáze soutěže, dojde opět k fázi spolupráce mezi jedinci, kde si vymění informace o tom, který z nich našel nejlepší řešení daného úkolu. Ostatní jedinci opustí své nalezené zdroje potravy a migrují (fáze

soutěžení) směrem k jedinci s nejlepším řešením a během této migrace se snaží nalézt ještě lepší řešení problému. Tento cyklus se opakuje do té doby, než se všichni jedinci sejdou u ještě lepšího řešení problému.

V daném principu vzniká řešení problému jedinců. Hledaný problém můžeme přirovnat k chování smečky lovících vlků, kteří hledají zdroje potravy (nalezení hledaného problému). I když je princip značně zjednodušen, je SOMA algoritmus schopný nalézt velmi hluboké a globální extrémy (zdroje potravy) u daných účelových funkcí (životních prostředí jedinců z populace). Souvislost mezi počítačovou implementací a biologickou realitou je popsána v tabulce:

Biologická realita	Počítačová implementace
členové smečky, společenství	jedinci v populaci, parametr NP
člen společenství s nejlepším zdrojem potravy	Leader, vedoucí daného migračního kola
potrava	vhodnost, hodnota účelové funkce, geometricky je to lokální či globální extrém na N rozměrné hyperploše
životní prostor společenství	Hyperplocha daná účelovou funkcí
migrace členů společenství v životním prostředí	Migrační kola v algoritmu SOMa

tabulka 3 – Význam biologické reality v algoritmu SOMA

Detailní popis funkce principu algoritmu SOMA:

1. **Definice parametrů.** Je nutné nadefinovat řídicí a ukončovací parametry (viz. Tabulka 2). Dále je nutné nadefinovat účelovou funkci, kterou budeme optimalizovat. Daná účelová funkce slouží pro budoucí jedince z populace jako pomyslné životní prostředí. Účelová funkce by měla vracet skalár, který bude použit jako měřítko kvality daného jedince.
2. **Tvorba populace.** Vytváří prvopočáteční populaci pomocí generátoru náhodných čísel. Díky Specimen a generátoru je pro každý parametr generováno náhodné číslo, jehož typ a hodnota je dána rozsahem v odpovídajícím prvku Specimenu.
3. **Migrační kola.** U každého jedince dojde k ohodnocení účelovou funkcí a je zvolen Leader (jedinec s nejlepší hodnotou (min., max.) účelové funkce) pro následující migrační kolo. V tomto okamžiku se začnou ostatní jedinci pohybovat směrem k Leaderovi pomocí skoků, jejichž velikost je dána parametrem Step. Jedinec si po každém skoku na nové pozici přepočítá svou hodnotu účelové funkce a pokud je daná hodnota lepší nežli předchozí, tak si ji zapamatuje. Tento pohyb jedince směrem k Leaderovi po skocích pokračuje tak dlouho, dokud není dosaženo pozice, která je

dána parametrem Mass. Po ukončení běhu se jedinec vrací zpátky na pozici, kde došlo k nalezení nejlepší hodnoty účelové funkce během jeho cesty. Z toho vyplývá, že po skončení aktuálního migračního kola jsou všichni jedinci, kromě Leadera, přemístěni. Předtím, nežli daný jedinec započne svou cestu směrem k Leaderovi, dojde k vygenerování sekvence náhodných čísel, jehož počet je roven parametru D. Daná čísla jsou porovnány s parametrem PRT. Pokud je n-té vygenerované číslo větší než PRT parametr, potom je n-tý parametr PRTVectoru nastaven na 0 a v opačném případě na 1. PRTVector má vliv na výsledný pohyb jedince. Jedná se o perturbaci nebo taky rušení jeho pohybu, což je ekvivalent mutací u jiných evolučních algoritmů. To vede k pohybu jedince v N-k rozměrném podprostoru, který je kolmý k původnímu mateřskému prostoru.

Migračním procesem můžeme rovněž myslet proces soutěživě-kooperativního chování. V průběhu soutěživé fáze, kdy se snaží každý jedinec nalézt co nejlepší extrém (zdroj potravy) vzhledem k ostatním jedincům. Jedinci tímto způsobem soutěží mezi sebou. Jakmile skončí jejich běh, tak si navzájem „sdělí“ své hodnoty účelové funkce a zvolí nového Leadera – vedoucího migračního kola.

4. **Naplnění ukončovacích parametrů – testování.** Posuzování, zda je rozdíl mezi Leaderem (nejlepší jedinec) a nejhorším jedincem menší nežli AcceptedError. Dále se kontroluje, zda byly vykonány migrační cykly v počtu, který je dán parametrem Migrate. V případě, že tato podmínka není splněna, se proces vrací na krok 3.
5. **Stop.** Nejlepší nalezené řešení po posledním migračním kole.

6.7 Strategie SOMA algoritmu

Pojem strategie v algoritmu SOMA znamená kooperaci jedinců a geometrické přesouvání populace po hyperploše (prostor možných řešení). Všechny verze jsou navzájem prakticky plně porovnatelné ve smyslu nalezení globálního extrému. Jedná se o strategie:

1. **AllToOne – Všichni k jednomu.** Všichni jedinci z populace migrují k Leaderovi.
2. **AllToAll – Všichni ke všem.** Zde neexistuje Leader. Všichni jedinci migrují ke všem ostatním tak jako ve verzi „Všichni k jednomu“ až na to, pokud je migrace aktuálního jedince dokončena, tak se daný jedinec vrací na pozici, kde byl nalezen nejlepší extrém

během jeho NP-1 migračních cest vykonaných během jednoho migračního kola. Přesto, že je tato strategie náročnější na výpočet dochází k vyšší pravděpodobnosti nalezení globálního extrému. To způsobuje fakt, že během migrace jedince se při této strategii prohledá větší část prostoru možných řešení.

3. **AllToAllAdaptive – Adoptivně všichni ke všem.** Rozdíl této strategie oproti strategii AllToAll spočívá v tom, že aktuálně migrující jedinec se nepřesouvá ke všem ostatním až po všech migracích, ale po každé aktuálně dokončené migraci ke každému z NP-1 jedinců se přesune na lepší pozici nalezenou na této aktuální migraci. Z této pozice se provádí migrace k dalším zbývajícím jedincům.
4. **AllToOneRand – Všichni k jednomu náhodně.** V této strategii se všichni jedinci pohybují k jednomu jedinci, tzv. Leaderovi, který není určen nejhlubší pozicí na hyperploše, ale je pro migraci každého jedince náhodně vybrán z populace. Jedinci pak nejsou vybírání náhodně, ale podle vhodnosti tak, jak tomu je u algoritmů genetických.
5. **Clusters – Svazky.** Jedná se o úpravu, kterou můžeme použít na jakoukoliv z předchozích strategií. Slovo „Svazek“ znamená, že jedinci účastníci se migračního procesu jsou rozděleni do tzv. svazků. V každém v tomto svazku pak probíhá samostatný SOMA algoritmus. Z důvodu pohybu jedinců se svazky mohou spojovat a rozpadat. Každý jedinec z populace je testován, do jakého svazku patří, podle vztahu:

$$CD > \sqrt{\sum_{i=1}^{\dim} \left(\frac{IND_i - CC_i}{HB_i - LB_i} \right)^2} \quad (5)$$

kde parametr IND_i je i -tý parametr jedince, CC_i je i -tý parametr Leadera (centrum svazku), HB_i a LB_i jsou povolené hranice jednotlivých parametrů (viz Specimen). CD je velikost svazku daná uživatelem. Výsledkem je tedy vytvoření svazků, které obsahují jedince populace. V případě, že některý z jedinců je příliš daleko ostatních a není schopen být zahrnut do již existujících svazků, je svazkem sám sobě a migruje ke všem ostatním jako ve strategii AllToAll.

6.8 Závislost SOMA algoritmu na řídicích a ukončovacích parametrech

U základní verze SOMA je počet ohodnocení účelové funkce dán vztahem:

$$N_{eval} = \frac{(Population - 1) * Mass * Migrations}{Step} \quad (6)$$

$$P_{GE} = \frac{N_{eval}}{L^n} \quad (7)$$

Dané vztahy vzniknou na základě principu, když se pohybuje jedinec směrem k vedoucímu daného migračního kola, pak na své dráze provede celkem Mass/Step skoků – dojde k ohodnocení účelové funkce. Populace o NP jedincích se pohybuje pouze NP-1 (vedoucí jedince je v klidu), potom je počet ohodnocení účelové funkce za jedno migrační kolo a celou populaci roven (NP-1)*Mass/Step. Počet ohodnocení za všechna migrační kola je dán rovnicí (6)

V případě zjednodušení je nutné předpokládat, že účelová funkce je definována v celočíselném oboru a z toho vychází konečný i počet hodnot, kterých může nabývat jak účelová funkce, tak její argumenty.

Vztah (7) určuje pravděpodobnost nalezení globálního extrému, kde n_{eval} je počet ohodnocení účelové funkce během optimalizace. Ze vztahu je dáno, že se pravděpodobnost zvyšuje zároveň s rostoucím čitatelem. Čítatel roste pouze, když se ze vztahu (7) zvětšuje parametr Mass nebo Migrations, nebo v případě že se zmenšuje parametr Step, nebo kombinací všech tří možností.

6.9 Obecné příklady na SOMA algoritmus

SOMA algoritmus slouží k řešení různorodých problémů, které se velmi často uplatňují v inženýrské praxi. Typickým příkladem může být třeba příklad, kdy máme tlakovou nádobu, která má uvnitř stlačený vzduch pracující s tlakem a minimálním objemem. Účelovou funkci zvolíme cenu za tlakovou nádobu, kde smyslem evoluční optimalizace je tuto cenu co nejméně snížit. Po zvolení jednotlivých proměnných parametrů nádoby zjistíme účelovou funkci $f(x)$, která reprezentuje finální výrobní cenu tlakové nádoby jako funkci parametrů nádoby. Toto je však pouze obecné naznačení, více se můžete dočíst v [1].

7 DIFERENCIÁLNÍ EVOLUCE

Jedná se o poměrně nový typ evolučního algoritmu (od r. 1995). Jeho schéma je dost podobné genetickým algoritmům, kterým se tento algoritmus podobá, jako například v tvorbě potomků, používání generací apod. Základní princip navržený R. Stornem spočíval ve vytváření populace potomků ve zvláštní populaci, která se účastní soupeření o místo v nové populaci až do té doby, než je zvláštní populace naplněna. Celý algoritmus se cyklicky opakuje v tzv. generacích (jedna generace = jeden cyklus) a pracuje s populacemi.

7.1 Parametry a terminologie

Stejně jako u algoritmu SOMA je činnost a kvalita Diferenciální evoluce ovlivněna jejími řídicími parametry. Jedná se o tyto následující parametry:

Řídicí parametr	Interval	Optimum?	Poznámka
NP	<2D, 100D?>	10D	Velikost populace, 100D pokud je funkce vysoce multimodální
F	<0, 2>	0,3-0,9	Mutační konstanta
CR	<0,1>	0,8-0,9	Prah křížení, CR << 1 jestli je funkce separabilní a CR=1 pokud je neseperabilní
Generations	Uživatel		Počet kol šlechtění populace

tabulka 4 – Parametry diferenciální evoluce

Označení a význam parametrů:

- **CR** – práh křížení. Jedná se o separabilní funkci, potom je nutné mít nastaven tento parametr na hodnoty blízké 0 nebo v opačném případě 1. Pokud je CR nastaven na 0, pak se mutace nedostane do zkušební jednotky, který díky tomu bude čistou kopií aktuálního (čtvrtého rodiče). V tomto případě dojde k zastavení vývoje evoluce. Pokud bude CR nastaven na 1, bude zkušební jedinec tvořen pouze ze tří náhodně vybraných rodičů – jedinců z populace a diferenciální evoluce se bude spíše podobat náhodnému hledání než evolučnímu algoritmu. Proto by CR neměl nabývat nikdy těchto mezních hodnot.
- **NP** – parametr udávající velikost populace. Hodnoty jsou získány pouze zkušeností s tímto algoritmem. NP parametr by neměl být menší než 4. Jedná se o minimální velikost populace, při které diferenciální evoluce ještě pracuje.

- **F** – mutační konstanta. Jeden z posledních řídicích parametrů diferenciální evoluce.
- **D** – dimenze problému. Udává počet argumentů účelové funkce. Tento parametr se získává s řešeným problémem a pokud ho chceme změnit, je nutno přeformulovat celý problém.
- **Generations** – udává počet evolučních cyklů (generací) během nichž se populace vyvíjí. Tento parametr musí mít logicky větší hodnotu než 0.

7.2 Populace

S populací se pracuje stejně jako u SOMA algoritmu a u ostatních evolučních algoritmů (viz. kap. 6.3).

7.3 Mutace

Diferenciální evoluce má oproti klasické evoluční teorii tu zvláštnost, že k vytvoření dalšího potomka je potřeba ne dvou, ale čtyř rodičů. Dále hrají důležitou roli tzv. mutace, která provádí tento princip. Pro každého jedince jsou náhodně vybráni tři další nestejní jedinci z populace (r_1 , r_2 , r_3). Dále se pak pomocí těchto jedinců vytvoří tzv. šumový vektor „ v “, který je vlastně mutací kombinace těchto tří náhodně vybraných rodičů. Mutace se provede tak, že se rozdíl dvou prvních náhodně vybraných rodičů vynásobí mutační konstantou „ F “ a výsledný vektor se přičte ke zbývajícimu třetímu. Princip je vyjádřen následujícím vzorcem:

$$v_j = x_{r_3,j}^G + F \cdot (x_{r_1,j}^G - x_{r_2,j}^G) \quad (8)$$

kde $x_{1x,j}^G$, $x_{2x,j}^G$, $x_{3x,j}^G$ je j -tý prvek náhodně vybraného jedince z populace, F je mutační konstanta a v_j je šumový vektor.

7.4 Křížení

U diferenciální evoluce dochází ke křížení až po mutačním procesu. Zatímco u genetických algoritmů je první vytvořen potomek křížením a pak teprve dochází k jeho mutaci. Proces křížení u diferenciální evoluce znamená, že se ze čtvrtého zatím nepoužitého rodiče a šumového vektoru vytvoří nový jedinec. Jedná o tzv. zkušební vektor. Vektor se vytvoří pomocí konstanty či prahu křížení CR a to tak, že se v cyklu vybírají

korespondující parametry ze čtvrtého a šumového jedince (oba první parametry, oba druhé parametry...) a pro každou takto vybranou dvojici se vygeneruje náhodné číslo. Pokud je číslo menší nežli CR, do příslušného parametru se ve zkušebním vektoru přesune parametr z jedince šumového vektoru a naopak.

7.5 Princip činnosti diferenciální evoluce

Hlavním úkolem diferenciální evoluce je v cyklech, které se nazývají „generace“, vyšlechtit co nejlepší populaci jedinců. Jedincem rozumíme hodnotu účelové funkce, která je spojena s každým jedincem. Při průběhu každé generace se provádí dané kroky:

1. **Stanovení parametrů** – parametry, na kterých závisí chod celé evoluce (viz. tabulka 4). Dále musíme nadefinovat prototyp jedince Specimen – tím rozumíme z jakých typů čísel se budou skládat jedinci (jedná se o datové typy např. Real a Integer)
2. **Tvorba populace** – rozumí se vytvoření populace pomocí vygenerování množiny jedinců (matice) podle prototypového (Specimen) vektoru. Každý jedinec musí mít jeden prvek navíc, a tím je hodnota účelové funkce.
3. **Započetí cyklu generace** – jedná se o cyklus, který zabezpečuje postupné evoluční šlechtění každého jedince z populace během každé generace. Dochází k postupnému vybírání jednoho jedince (aktivní jedinec, cílový vektor) za druhým až do konce populace (ukončení jedné generace) a pro každého z nich je proveden následující evoluční cyklus.
4. **Evoluční cyklus** – během tohoto cyklu se provádí mutace a křížení, tzn. že se náhodně zvolí tři další různé vektory (jedinci) z populace. První dva se od sebe odečtou a vznikne tzv. „diferenční vektor“. Po vynásobení mutační konstantou „F“, která vektor zmutuje se získá „váhovaný diferenční vektor“. Ten se přičte k třetímu náhodně vybranému vektoru a vznikne „šumový vektor“. Poté se připraví tzv. „zkušební vektor“ a z cílového a šumového vektoru se bere postupně jeden prvek za druhým (první z obou, druhý z obou...) a pro vybranou každou dvojici je generováno náhodné číslo v rozsahu 0-1 a je porovnáváno s konstantou CR. Pokud je dané číslo menší než CR, pak se do příslušné pozice v tzv. „zkušebním vektoru“ umístí prvek z vektoru šumového a v opačném případě z vektoru cílového. Touto operací vznikne zkušební vektor, jehož hodnota účelové funkce je porovnána s hodnotou účelové funkce

cílového vektoru. Na pozici cílového vektoru v nové populaci je vybrán ten vektor – jedinec, který má hodnotu účelové funkce lepší.

$$x_{i,j}^{zkušební} = \begin{cases} x_{r3,j}^G + F \cdot (x_{r1,j}^G - x_{r2,j}^G) & \text{jestli } rand_j(0,1) \leq CR \vee j = k \\ x_{i,j}^G & \text{jinak} \end{cases} \quad (9)$$

kde

$i=1,\dots, NP, j=1,\dots, D$

$k \in \{1,\dots, D\}$, náhodný index

$r_1, r_2, r_3 \in \{1,\dots, NP\}$, náhodný vývěr tří dalších jedinců když:

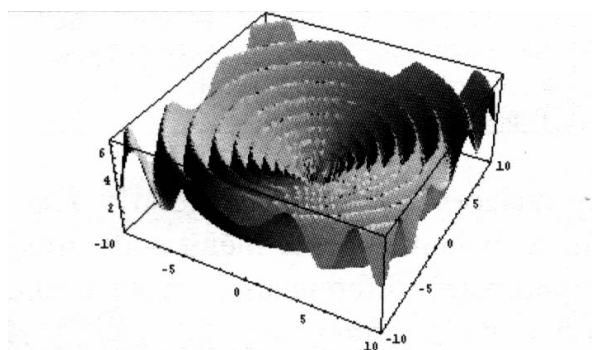
$r_1 \neq r_2 \neq r_3 \neq i$

$CR \in < 0,1 >, F \in < 0,1 >$

$$X_i^{G+1} = \begin{cases} x^{zkušební} & \text{jestli } f_{\cos t}(x^{zkušební}) \leq f_{\cos t}(X_i^G) \\ X_i^G & \text{jinak} \end{cases} \quad (10)$$

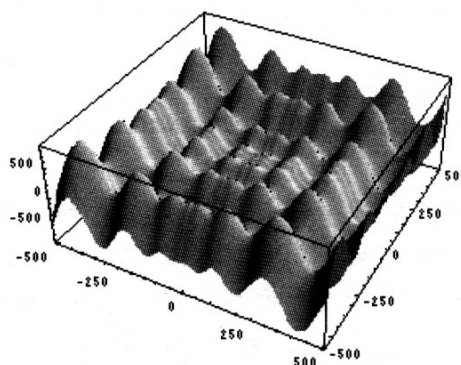
5. **Testování naplnění ukončovacích parametrů.** K ukončení diferenciální evoluce dojde pouze tehdy, provede-li se uživatelem zadaný počet generací. Tento princip je jediný ukončovací parametr tohoto algoritmu. Může dojít k výjimečným případům, kdy například pokud se během posledních dvaceti generací nezměnila hodnota nejlepšího jedince, tak dojde k ukončení procesu.
6. **Vyhodnocení** – celý proces generací se opakuje, dokud není vyčerpán zadaný počet generací. V průběhu jedné generace dojde k uchování hodnoty účelové funkce nejlepšího jedince do vektoru historie, který po ukončení znázorňuje průběh celého evolučního procesu. Při vyhodnocování hodnoty účelové funkce po každé generaci je možné přidat algoritmy, které evoluci ukončí, pokud se hodnota účelové funkce nejlepšího jedince po dobu posledních např. deseti generací nezlepšila. Také je možné při takovéto stagnaci proces evoluce neukončovat, ale pouze za chodu změnit parametry procesu (F, CR), což může také způsobit opětovnou konvergenci účelové funkce k nižším hodnotám. Kvalitu a průběh šlechtění lze ovlivnit mnoha faktory:
 - **nastavením řídicích parametrů** – kombinací řídicích parametrů může mít vliv na průběh evoluce. Může vznikat méně nových jedinců, což odpovídá nízké hodnotě CR

- **velikostí populace** – když bude populace malá, tak bude horší výběr a při velké populaci bude potřeba více času na vytvoření nové
- **počtem generací** – v případě, že zadáme malý počet generací, tak evoluce může skončit dříve než nalezne extrém. Z toho plyne pravidlo, že více populací je lepší než méně.
- **definicí účelové funkce** – v případě špatně nadefinované účelové funkce může evoluce probíhat pomalu nebo nemusí probíhat vůbec
- **definicí intervalu** – jedná se o interval, který je definován při evoluci a ve kterém se má evoluce uskutečnit. Pokud evoluce poběží na funkci, která nebude omezená intervalem, potom dojde bez problému k nalezení extrému – extrémy se „koncentrují“ okolo počátku a evoluce tím pádem bude konvergovat k těmto hodnotám (Obr. 3).



Obr. 3- Ukázka neomezené funkce

Pokud použijeme funkci na Obr. 4, pak se evoluce zastaví až v nekonečnu. To je příčinou zvětšujících se hodnot extrému s rostoucí vzdáleností od počátku.



Obr. 4 – Ukázka omezené funkce

V případě že jsou nejasnosti o tvaru funkce, na které je hledán extrém, je vždy lepší definovat interval, na kterém se evoluce uskuteční. Z toho plyne pokud definujeme omezující hranice jednotlivých parametrů tak se evoluce drží uvnitř prohledávaného prostoru možných řešení

Diferenciální evoluce se odlišuje oproti jiným algoritmům tím, že je potřeba ne dvou, ale celkem čtyř rodičů, kteří vytvoří potomka. Tento potomek nakonec soupeří o místo v nové populaci. K tvorbě potomka dochází dvojím způsobem:

- pomocí **křížící konstanty CR** (jako u jiných evolučních algoritmů)
- pomocí **šumového vektoru**

Díky těmto dvou způsobům se z diferenciální evoluce stává poměrně robustní algoritmus s poměrně vysokou diverzibilitou, což znamená že diferenciální evoluce vybírá rodiče náhodně a díky tomu vždy nalezne globální extrém.

7.6 Varianty diferenciální evoluce

V těchto možných variantách je ukázáno deset různých způsobů, jak se vypočítá šumový vektor „v“, který lze dosadit do (9) místo aktuálně zobrazeného šumového vektoru „v“, vypočteného podle Varianty 7 –De/rand/1/bin, jedné z možných deseti variant:

Varianta 1: DE/best/1/exp

$$v = x_{best,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G) \quad (11)$$

Varianta 2: DE/rand/1/exp

$$v = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G) \quad (12)$$

Varianta 3: DE/rand-to-best/1/exp

$$v = x_{i,j}^G + \lambda \cdot (x_{best,j}^G - x_{i,j}^G) + F \cdot (x_{r1,j}^G - x_{r2,j}^G) \quad (13)$$

Varianta 4: DE/best/2/exp

$$v = x_{best,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G) \quad (14)$$

Varianta 5: DE/rand/2/exp

$$v = x_{r5,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G) \quad (15)$$

Varianta 6: DE/best /1/exp

$$v = x_{best,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G) \quad (16)$$

Varianta 7: DE/rand /1/bin

$$v = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G) \quad (17)$$

Varianta 8: DE/rand-to-best/1/bin

$$v = x_{i,j}^G + \lambda \cdot (x_{best,j}^G - x_{i,j}^G) + F \cdot (x_{r1,j}^G - x_{r2,j}^G) \quad (18)$$

Varianta 9: DE/best/2/bin

$$v = x_{best,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G) \quad (19)$$

Varianta 10: DE/rand/2/bin

$$v = x_{r5,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G) \quad (20)$$

kde x_{best} představuje nejlepšího jedince z populace. Příčinou, že pořadí $r1$, $r2$ a $r3$ je v (9) a (17), je způsobeno tím, že vektory vytvářející šumový vektor jsou vybírány náhodně. Důvod toho, že některé varianty jsou stejné je způsoben tím, že zde jsou vypsány pouze vlastní postupy výpočtu vektoru „v“.

7.7 Stagnace

Jednou z „nevýhod“ diferenciální evoluce je tzv. stagnace. Stagnace je typická pro tento algoritmus. Jedná se o jev, při kterém dochází k zastavení vývoje hodnoty účelové funkce směrem k nižším hodnotám ještě před dosažením globálního extrému. Tento jev se projevuje bez zřejmých důvodů a na rozdíl od předčasné konvergence k suboptimálnímu řešení tak, že populace stále zůstává diverzibilní zatímco optimalizační proces dále nepokračuje. K předčasné konvergenci k suboptimálnímu řešení dochází tehdy, pokud:

- optimalizační proces zavedl populaci do lokálního extrému dané účelové funkce. To znamená, že parametry jedinců se od sebe skoro neliší a populace ztratila svou rozmanitost.
- populace ztratila diverzibilitu
- optimalizační proces probíhá pomalu nebo neprobíhá vůbec

Na rozdíl od těchto obecných poznatků optimalizační proces běžící na bázi diferenciální evoluce vykazuje za určitých podmínek stagnaci navzdory faktu, že:

- populace nezkonvergovala do lokálního extrému
- populace neztratila diverzibilitu
- optimalizační proces dále neprobíhá navzdory faktu, že vznikají noví jedinci.

Tato situace se nazývá stagnací. Stagnace je tedy jev, který může nastat při vhodné velikosti populace a řídicích parametřů v diferenciální evoluci. Tyto poznatky rovněž využijeme při návrhu populace a řídicích parametřů u diferenciální evoluce.

K vygenerování nové populace použijeme (9) a (10), přičemž na základě principů diferenciální evoluce zde existuje konečný počet možných zkušebních řešení (vektorů), které nesou označení n_{trial} . Všechna možná řešení v podstatě představují kandidáty do nové populace. Zde je vidět, že riziko stagnace je úměrné počtu nových možných řešení, nových jedinců – kandidátů, které algoritmus vygeneruje během jedné generace. Z toho vyplývá, že čím větší je počet možných řešení během jedné generace, tím je menší riziko stagnace. Počet řešení pro verzi diferenciální evoluce označovanou jako (17) je dán vztahem:

$$n_{trial} = \begin{cases} NP^3 - 3 \cdot NP^2 + 2 \cdot NP & \text{jestli } CR = 1 \\ D \cdot NP \cdot (NP^3 - 3 \cdot NP^2 + 2 \cdot NP) & \text{jestli } CR = 0 \\ 2^D \cdot NP \cdot (NP^3 - 3 \cdot NP^2 + 2 \cdot NP) & \text{jinak} \end{cases} \quad (21)$$

Aktuální počet různých zkušebních řešení závisí na parametru F a aktivní populaci. Faktory ovlivňující riziko stagnace jsou NP, F, CR a daná účelová funkce, která je dána, takže uživatel může ovlivnit jen parametry NP, F a CR.

7.7.1 Vliv mutace F

Vektor mutace vytváří tvorbu nových řešení významným způsobem. Při F=1.0 dochází ke generování dvojic stejných řešení, což podporuje riziko stagnace. Proto je možná dobré zvolit F=0,99, pokud je potřeba velkého faktoru mutace.

7.7.2 Vliv křížící konstanty CR

V případě nastavení CR=1,0 nenarůstá počet nových řešení, protože novým kandidátem se stává šumový vektor. Při nastavení CR=0,99 dochází k rapidnímu zvýšení možnosti tvorby nových kandidátů – vektorů. Z tohoto důvodu není dobré nastavit hodnotu CR=1, ale těsně pod ní v případě, kdy je potřeba silné mutace.

7.7.3 Vliv velikosti populace NP

Při zvětšování populace se silně redukuje riziko stagnace a předčasné konvergence. Při růstu velikosti populace NP vzrůstá počet možných nových řešení. Tím ale však dochází ke zpomalení rychlosti nalezení globálního extrému.

7.7.4 Stagnace- doporučená nastavení parametrů

Operace reprodukce (4) je schopna vygenerovat jen konečný počet potenciálních zkušebních řešení (5). Z toho důvodu pokud žádné z nich není schopno nahradit řešení stávající, potom dochází ke stagnaci, která tím pádem závisí na počtu nových zkušebních řešení, tak také na jejich schopnosti získat místo v nové populaci. Závisí tedy na:

- Velikosti populace NP - větší populace snižuje riziko, většinou stačí $NP > 20$
- Mutaci F – není doporučeno nastavit $F=1.0$ z důvodu vysokého rizika vzniku stagnace
- Křížící konstantě CR – rovněž není doporučeno nastavit $F=1.0$ z důvodu vysokého rizika stagnace
- Aktuální populaci
- Účelové funkci

7.8 Obecné příklady na diferenciální evoluci

V [1] je uvedena celá řada příkladů na diferenciální evoluci, tak i SOMA algoritmus. Příklad, který zde naleznete se zabývá návrhem pružiny. Cílem optimalizace je nalézt množství oceli potřebné k vytvoření pružiny. Proměnné jsou zde charakterizovány jako počet závitů pružiny, vnější průměr a průměr prutu pružiny.

8 SROVNÁNÍ KLASICKÝCH A MODERNÍCH OPTIMALIZAČNÍCH METOD

V této kapitole budu porovnávat klasické a moderní optimalizační metody z různých pohledů. Zřejmé je, že každá z metod nachází uplatnění v různých oborech lidské činnosti. Operační výzkum a evoluční algoritmy vycházejí ze základního jádra, ale dále už se naprosto odlišují. I když jsem ve své práci důkladně rozebíral algoritmy SOMA a diferenciální evoluci, přesto budu hovořit o moderních optimalizačních metodách jako evolučních algoritmech.

8.1 Účelová funkce

Ať se jedná o metodu klasickou či moderní, účelová funkce bude vždy geometrický problém, v jehož rámci se hledá nejnižší či nejvyšší pozice na N rozměrné ploše. Z toho vyplývá, že vždycky musíme dojít k nějakému výsledku, tzv. cíli, který má podobu maxima či minima.

8.1.1 Operační výzkum

V případě operačního výzkumu vycházíme vždy z reálných předpokladů, že chceme něco minimalizovat (např. náklady) nebo maximalizovat (např. zisk.) na určité množině možných variant, které určujeme soustavou lineárních nebo nelineárních rovnic. V případě firmy se jedná o výrobní program, který přináší užitek o velikost účelové funkce $f(x)$.

8.1.2 Evoluční algoritmy

U evolučních algoritmů je princip účelové funkce podobný, ne-li stejný. V těchto případech se však účelová funkce řeší jako geometrický problém, kde se rovněž hledá maximum či minimum dané funkce na N rozměrné ploše. Tuto plochu označujeme jako hyperplochu či prostor možných řešení.

8.2 Podmínky omezení

Vždycky musíme brát v úvahu prostor, na kterém se bude daná metoda či algoritmus uskutečňovat. Nikdy není možné, že by prostor byl neomezený, vždycky je dán omezujícími podmínkami.

8.2.1 Operační výzkum

Vycházíme vždycky z toho, že operace se uskutečňují stejně jako u evolučních algoritmu na definovaném prostoru. Tento prostor však zde nemá podobu hyperplochy jako u evolučních algoritmů, ale rozumíme jím pevně zadané podmínky v podobě stanovení cíle analýzy, popisu procesů a popisem činitelů. Z toho vyplývá, že proces v podobě výroby výrobků se v podniku vyrábí s určitou intenzitou, která má vliv na cíl analýzy a činiteli se rozumí, že výrobky se nemůžou vyrábět neomezeně a k jejich realizaci je potřeba celá řada činitelů, ať už se jedná o spotřebu času na výrobek či spotřebu surovin. Cílem analýzy je u matematického modelu vyjádřen jako lineární funkce $z=f(x)$, kde musíme nalézt extrém (maximum či minimum). Tato funkce však nemusí být vždy lineární, ale operační výzkum není schopný řešit příliš komplikované úlohy.

8.2.2 Evoluční algoritmy

Evoluční algoritmy jsou dány omezujícími podmínkami v podobě populace, která má v sobě jedince. Každý jedinec nese řešení problému v podobě množiny argumentů účelové funkce a je s ním navíc spojena i hodnota účelové funkce. Argumenty účelové funkce (parametry jedince) mohou nabývat pouze určitých hodnot. S tím také souvisí nutnost vhodného ošetření těchto argumentů v průběhu evolučního procesu. Při řešení optimalizačních problémů, ať už v inženýrské praxi, v ekonomice či v oblasti aplikované informatiky, dochází vždy k nalezení optima zadané účelové funkce, kde se argumenty této funkce mohou pohybovat pouze v určených mezích a mohou nabývat pouze určitých hodnot.

8.3 Základní stanovení podmínek

K tomu, abychom dosáhli cíle optimalizace musíme stanovit počáteční podmínky ze kterých vycházíme po celou dobu provádění daného procesu.

8.3.1 Operační výzkum

V operačním výzkumu vycházíme z proměnných, které jsou prvním předpokladem k úspěšnému sestavení matematického modelu. Tyto proměnné mají většinou podobu procesů probíhajících ve výrobě. Tyto procesy jsou však stejně jako u evolučních algoritmů

ovlivněny parametry, které jsou nezbytné při sestavování matematického modelu. Mějme podnik ve kterém probíhá 5 procesů a je ovlivněn těmito parametry (činiteli):

	Noha k židli	Noha ke stolu	Stůl	Židle 1	Židle 2
Dřevo [kg/ks]	0,3	0,8	5	1,5	2
Pracnost [hod/ks]	0,25	0,4	2	1,5	2,5

tabulka 5 – Ukázka počtu proměnných a počtu činitelů v reálném příkladě

Jak je z tabulky vidět, tak podnik provádí 5 procesů (5 proměnných) a je omezen 2 parametry (2 činitelé). V tomto případě to znamená, že výroba polotovarů a hotových výrobků v podniku je omezena spotřebou dřeva a dobou, za kterou se výrobek(nebo polotovar) vyrobí, tzv. pracností.

8.3.2 Evoluční algoritmy

V případě evolučních algoritmů je situace podobná jako u operačního výzkumu. Populace je tvořena maticí o rozměrech NP x D. (viz. kap. 6.2 a 6.3). Tato matice se skládá z jedinců a počtu parametrů (optimalizovaných proměnných nebo také počtu argumentů účelové funkce).

	Jedinec	Jedinec 2	Jedinec 3	Jedinec 4	Jedinec 5	Jedinec 6	Jedinec 7
CV	204,95650	273,56466	163,47697	158,46466	110,46466	108,44846	188,83466
Parametr 1	3,06178	-46,46767	5,76768	40,77667	35,76768	0,07468	23,76768
Parametr 2	2,51176	57,74676	85,46768	0,29468	24,46767	4,28465	20,38465
Parametr 3	46,46467	51,47677	11,76977	3,08747	5,88846	58,74677	33,74646
Parametr 4	78,46146	16,46767	2,76977	3,85465	12,76768	4,58888	4,76765
Parametr 5	7,46465	57,67676	58,46977	28,46767	4,87786	23,76976	4,22713
Parametr 6	73,46467	-40,46767	0,57765	50,76768	4,88777	30,76768	40,50507

tabulka 6 – Ukázka populace s jedinci a argumenty účelové funkce

U operačního výzkumu jsou parametry, které podnik potřebuje k sestavení matematického modelu závislé na výrobních vlastnostech podniku. V případě evolučních algoritmů tyto hodnoty jsou získány vygenerováním hodnot náhodných čísel.

8.4 Shrnutí

V těchto základních bodech jsem shrnul v čem jsou klasické a moderní optimalizační podobné.. Dále se pak každá z metod ubírá jinými směry. V případě operačního výzkumu se na základně stanovených proměnných dopočítáme k celkovému výsledku, který vyjadřuje optimální výrobní program. U evolučních algoritmů probíhá proces s populací jedinců, který je u každého z algoritmů jiný (viz. kap. 6 a 7).

ZÁVĚR

Rozsah optimalizačních metod je příliš veliký, ne-li obrovský. Vzhledem k tomu, že optimalizační metody nalézají uplatnění ve všech různých oborech lidské činnosti, je o nich vydána řada publikací. Cílem této práce bylo:

- uvést čtenáře do základních problémů v oblasti optimalizace
- sestavit přehled typů optimalizačních metod a jejich zařazení do příslušné skupiny, nastínit několik základních metod z oblasti optimalizace
- provést srovnání mezi probranými metodami a zhodnotit výsledky

Všechny cíle jsem průběžně plnil v celém textu práce. V úvodní kapitole jsem nastínil vývoj optimalizačních metod a v dalších kapitolách jsem prozkoumal tyto metody podrobně. V předposlední kapitole jsem provedl jejich srovnání a došel jsem k výsledným závěrům.

Zjistil jsem, že všechny metody vedou k výslednému cíli, a to nalézt optimální řešení. Je jedno jakou podobu toto řešení má. Vždycky se bude lišit a pokaždé závisí na definičním oboru (omezujičích podmínkách), ve kterém dané řešení hledáme. Abychom však k tomuto řešení došli, je nutné stanovit počáteční podmínky ze kterých vycházíme. Úkol musí být pokaždé vyhodnocen tak, aby došlo k nalezení nějakého extrému (minima či maxima).

Myslím si, že perspektiva optimalizačních metod má velkou budoucnost. Budou vymýšleny stále nové algoritmy, které se při zrychlování technického vývoje budou uskutečňovat stále rychleji a budou sloužit k řešení stále složitějších problémů. Tyto problémy však budou vždycky vycházet ze stejných pravidel.

Z důvodu, že tato práce byla napsána jako obecný přehled problematiky, jsem zde neuvedl počítání příkladů, což by bohužel bylo nad rozsah této práce. Snad budou stačit pouhé teoretické principy popsané v kapitolách. Z toho důvodu jsem alespoň uvedl principy výpočtů SOMA algoritmů a diferenciální evoluce do přílohy, aby čtenář získal přehled pro provádění principů algoritmů v praxi.

Pevně doufám, že tato bakalářská práce poslouží některým čtenářům jako základní příručka pro orientaci v klasických a moderních optimalizačních metodách. V případě, že by někdo z nich chtěl mít větší zájem o toto téma, nezbyvá mi nic jiného, než je odkázat na seznam použité literatury.

SEZNAM POUŽITÉ LITERATURY

- [1] Zelinka, Ivan Umělá inteligence v problémech optimalizace, BEN, 2002, 190 p.
ISBN 80-7300-069-5
- [2] Jablonský, J.:Operační výzkum, Kvantitativní modely pro ekonomické rozhodování, PROFESSIONAL PUBLISHING 2002, ISBN 80-86419-42-8
- [3] Kvasnička, V., Pospíchal, J., Tiňo, P.:Evolučné algoritmy, STU v Bratislavě, 2002, ISBN 80-227-1377-5
- [4] Kaňka, M., Henzler, J.: Matematická analýza (Matematika B pro VŠE), VŠE v Praze, fakulta informatiky a statistiky, 1995

SEZNAM OBRÁZKŮ

OBR. 1 - FÁZE PŘI APLIKACI OPERAČNÍHO VÝZKUMU	18
OBR. 2 – UKÁZKA JEDNODUCHÉHO GRAFU	22
OBR. 3- UKÁZKA NEOMEZENÉ FUNKCE	41
OBR. 4 – UKÁZKA OMEZENÉ FUNKCE	41

SEZNAM TABULEK

TABULKA 1 – POROVNÁNÍ KLASICKÝCH A MODERNÍCH OPTIMALIZAČNÍCH METOD	10
TABULKA 2 – PARAMETRY SOMA ALGORITMU	29
TABULKA 3 – VÝZNAM BIOLOGICKÉ REALITY V ALGORITMU SOMA.....	33
TABULKA 4 – PARAMETRY DIFERENCIÁLNÍ EVOLUCE	37
TABULKA 5 – UKÁZKA POČTU PROMĚNNÝCH A POČTU ČINITELŮ V REÁLNÉM PŘÍKLADĚ.....	48
TABULKA 6 – UKÁZKA POPULACE S JEDINCI A ARGUMENTY ÚČELOVÉ FUNKCE	48

SEZNAM PŘÍLOH

- P I Princip výpočtu algoritmu SOMA.
- P II Princip výpočtu diferenciální evoluce.

PŘÍLOHA P I: PRINCIP VÝPOČTU ALGORITMU SOMA.

Parametry SOMA

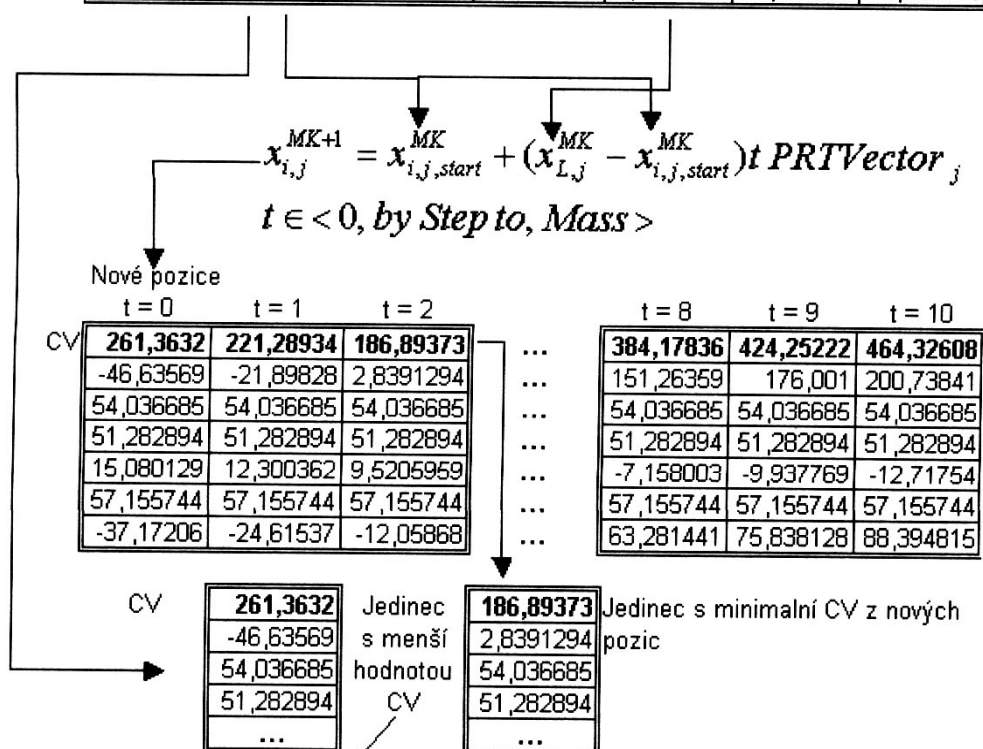
Step	0,3
Mass	3
PRT	0,1
AcceptedError	0,1
Migrations	1000
NP	7

PRT vektor, pro běh každého jedince je jiný

If Rand < PRT then 1 else 0	↔	1
If Rand < PRT then 1 else 0	↔	0
If Rand < PRT then 1 else 0	↔	0
If Rand < PRT then 1 else 0	↔	1
If Rand < PRT then 1 else 0	↔	0
If Rand < PRT then 1 else 0	↔	1

Účelová funkce $f(\mathbf{x}) = \text{Abs}(\text{Parametr 1}) + \text{Abs}(\text{Parametr 2}) + \dots + \text{Abs}(\text{Parametr 5})$

	Aktivní jedinec		Leader				
	Jedinec 1	Jedinec 2	Jedinec 3	Jedinec 4	Jedinec 5	Jedinec 6	Jedinec 7
CV	204,91528	261,3632	163,79679	121,73019	107,52784	121,06024	120,20974
Parametr 1	3,0615753	-46,63569	5,0246553	38,723912	35,822343	0,0715185	23,761224
Parametr 2	2,5117282	54,036685	85,104704	0,2928606	24,111443	4,2879691	20,384665
Parametr 3	46,75014	51,282894	11,347164	3,0796963	24,657689	60,241731	33,437248
Parametr 4	72,486617	15,080129	2,916686	3,6713463	5,8142407	4,5385164	4,0482021
Parametr 5	6,316564	57,155744	58,829537	26,610056	12,43856	23,891907	4,2271271
Parametr 6	73,788657	-37,17206	0,5740442	49,352316	4,6835676	28,028598	34,351273



	Jedinec 1	Jedinec 2	Jedinec 3	Jedinec 4	Jedinec 5	Jedinec 6	Jedinec 7
CV	204,91528	186,89373					
Parametr 1	3,0615753	2,8391294					
Parametr 2	2,5117282	54,036685					
Parametr 3	46,75014	51,282894					
Parametr 4	72,486617	9,5205959					
Parametr 5	6,316564	57,155744					
Parametr 6	73,788657	-12,05868					

PŘÍLOHA P II: PRINCIP VÝPOČTU DIFERENCIÁLNÍ EVOLUCE.

