Evolučné algoritmy

Riešenie úloh

Marián Mach

Košice

2013

doc. Ing. Marián Mach, CSc.
Katedra kybernetiky a umelej inteligencie
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach
Marian.Mach@tuke.sk

Táto práca bola vytvorená realizáciou projektu Rozvoj Centra informačných a komunikačných technológií pre znalostné systémy (kód ITMS projektu: 26220120030) na základe podpory operačného programu Výskum a vývoj financovaného z Európskeho fondu regionálneho rozvoja.

Edícia vedeckých spisov Fakulty elektrotechniky a informatiky TU Košice.

Lektorovali: *prof. Ing. Vladimír Olej, CSc., FES, Univerzita Pardubice*
Ing. Marek Bundzel, PhD., FEI, Technická univerzita v Košiciach

ISBN 978-80-553-1445-7

Bol som obeťou radu náhod, tak ako my všetci.

Kurt Vonnegut, Jr.

Predslov

Evolučné algoritmy sa postupne presunuli z pozície exotickej hračky na pozíciu štandardne použiteľnej riešiacej metódy. Stali sa bežným nástrojom pre riešenie optimalizačných a konštrukčných úloh, či už v numerickej alebo kombinatorickej oblasti. Uplatnenie nachádzajú najmä v prípadoch, keď nie je nutné nájsť optimálne riešenie ale postačujúcim je aj kvalitné suboptimálne riešenie.

Úroveň povedomia o tejto triede algoritmov v poslednom období vzrástla a to aj vďaka väčšej dostupnosti odbornej literatúry. Už nie je nemožnosťou natrafiť aj na publikácie v slovenskom jazyku. Skutočnosťou však ostáva to, že väčšina dostupnej literatúry sa venuje základným otázkam spojeným s evolučnými algoritmami. Výsledkom je absencia informácií o pokročilejších vlastnostiach a spôsoboch použitia týchto algoritmov.

Túto medzeru sa snaží vyplniť predkladaná publikácia. Svojim obsahom nadväzuje na základné informácie o evolučných algoritmoch a preto tieto informácie nevysvetľuje ale naopak predpokladá ich znalosť. Od čitateľa sa očakávajú znalosti problematiky evolučných algoritmov minimálne v rozsahu základnej štruktúry algoritmu, spôsobe jeho činnosti, realizácie základných funkčných blokov (ako inicializácia, selekcia, genetické operátory, vytváranie novej generácie a pod.) a nastavovania hodnôt parametrov algoritmu[1]. Aj keď to nie je podmienkou pre porozumenie predkladaného textu, výhodou sú aj znalosti o metódach udržiavania rôznorodosti a potláčania prílišnej konvergencie.

Cieľom je priblížiť čitateľovi možnosti použitia evolučných algoritmov pre riešenie rozličných typov úloh. Iba málo dostupných zdrojov sa orientuje na riešenie praktických úloh rôzneho typu. Typicky záujemca o problematiku po preštudovaní niektorého zo zdrojov v snahe použiť nadobudnuté znalosti pri riešení nejakej úlohy postupuje nasledovným spôsobom:

[1] Pre získanie potrebných informácií možno doporučiť napríklad [26], ale je možné ich nájsť aj v iných zdrojoch.

- rozhodne sa akým spôsobom budú reprezentované potenciálne riešenia, ktoré je potrebné prehľadať a spomedzi ktorých je potrebné hľadané riešenie vybrať,
- určí spôsob výpočtu kvality potenciálneho riešenia vzhľadom na jeho schopnosť riešiť uvažovaný problém,
- zvolí celkovú štruktúru evolučného algoritmu a realizáciu jednotlivých stavebných blokov,
- nastaví hodnoty parametrov vytvoreného algoritmu v snahe aby realizácia behu algoritmu bola schopná poskytnúť čo najkvalitnejšie riešenie v čo najkratšom čase.

Problémom je to, že bežný text o evolučných algoritmoch vybaví čitateľa zásobou znalostí, ktoré mu umožnia zostaviť "klasickú" verziu algoritmu vhodnú iba pre obmedzenú triedu úloh (typicky hľadanie jedného riešenia statického jednokriteriálneho problému bez dodatočných ohraničení). V praxi sa však veľmi často vyskytujú úlohy, ktoré nezapadajú do tejto jednoduchej triedy. A aj keď sa evolučné algoritmy dajú použiť aj pre takéto úlohy, pre ich úspešné použitie sú potrebné dodatočné znalosti ako obohatiť základný tvar evolučného algoritmu pre tú ktorú špecifickú triedu úloh.

Dôraz predkladaného textu je na štyroch triedach v praxi sa často vyskytujúcich úloh:

- multimodálne úlohy, pri riešení ktorých je dôležité nájsť nielen jedno riešenie ale všetky riešenia alebo určitý stanovený počet riešení,
- viackriteriálne úlohy, pri riešení ktorých je potrebné zohľadňovať splnenie viacerých kritérií, ktoré sú často kontradikčné,
- nestacionárne úlohy s dynamickými zmenami, pri ktorých je potrebné riešenie nielen jednorazovo nájsť ale následne aj monitorovať a trasovať jeho zmeny,
- úlohy s ohraničeniami, kladúcimi dodatočné požiadavky na prípustnosť hľadaného riešenia.

Kniha je v zásade rozdelená do štyroch kapitol. Každá kapitola je venovaná jednej z uvedených tried, pričom všetky kapitoly zdieľajú spoločnú štruktúru, pozostávajúcu z troch častí. Prvá časť predstavuje príslušnú triedu úloh, ktorej je venovaná daná kapitola, spolu s problémami, ktoré prináša

snaha použiť klasickú verziu evolučného algoritmu pre riešenie úloh danej triedy. Druhá časť sa venuje modifikáciám a rozšíreniam klasickej podoby evolučného algoritmu, umožňujúcich algoritmu prispôsobiť sa špecifikám danej triedy úloh. Je to vlastne katalóg metód vhodných pre špecifickú triedu úloh. Tretia časť je venovaná porovnaniu metód predstavených v predchádzajúcej časti. Porovnanie je empirické na základe riešenia vybranej testovacej úlohy. Keďže kniha si nerobí nárok na absolútne potvrdenie alebo zavrhnutie žiadnej z prezentovaných metód, cieľom empirického porovnania je skôr poskytnúť čitateľovi informáciu o použití jednotlivých metód v konkrétnom kontexte, ktorá mu môže uľahčiť voľbu pri riešení konkrétnej úlohy.

Vďaka organizácii knihy nie je nutné jej sekvenčné čítanie. Jednotlivé kapitoly, venované rozdielnym triedam úloh, sú na sebe nezávislé a je teda iba na výbere čitateľa, akú cestu cez predkladaný materiál zvolí.

Kniha je vhodná pre tých záujemcov, ktorí by chceli získať základný prehľad o možnosti použitia evolučných algoritmov pre riešenie úloh patriacich do jednej z preberaných tried. Rovnako je kniha vhodná pre tých, ktorí majú hlbší záujem o problematiku evolučných algoritmov a radi by si rozšírili svoje znalosti. Keďže však pre pochopenie predkladaného obsahu vyžaduje určitú úroveň znalostí o problematike, nie je vhodnou literatúrou pre čitateľov bez predchádzajúceho kontaktu s oblasťou evolučných algoritmov.

Kniha vznikla na Katedre kybernetiky a umelej inteligencie Fakulty elektrotechniky a informatiky Technickej univerzity v Košiciach. Autor pri jej písaní vychádzal zo svojich dlhoročných skúseností, získaných počas riešenia rôznych úloh ako aj zabezpečovania výuky predmetu Evolučné algoritmy na inžinierskom stupni štúdia. Preto je čiastočne aj dielom mnohých študentov tohto predmetu, ktorí svojimi podnetmi prinútili autora premýšľať, ako vykladať danú problematiku čo najprístupnejším spôsobom.

Na tomto mieste by som chcel vyjadriť poďakovanie všetkým tým mojim študentom, ktorí v rokoch 2008 až 2011 prispeli k empirickému porovnaniu prezentovaných metód. Poďakovanie patrí aj recenzentom za starostlivé prečítanie rukopisu, opravu formálnych aj vecných chýb a za cenné pripomienky a námety, ktoré obohatili úroveň predkladaného textu.

Košice, jún 2013 autor

Obsah

Kapitola 1

Hľadanie viacerých riešení

V prípade multimodálnych funkcií (funkcií majúcich viac extrémov) môže byť z aplikačného hľadiska zaujímavé nájsť nielen jedno (globálne) riešenie, ale buď všetky alebo aspoň určitý počet rôznych riešení. Príkladom je situácia, keď vhodnosť jedincov neodráža všetky požiadavky na riešenie, či už z dôvodu absencie ich explicitného vyjadrenia alebo obtiažnej formalizácie do tvaru funkcie vhodnosti – a teda je prijateľnejšie prostredníctvom evolučného algoritmu vyhľadať viacero riešení a konečný výber ponechať na človeka.

Preto je možné sa často stretnúť s úlohou vyhľadať viacero riešení. Zdalo by sa, že to je možné realizovať aj pomocou štandardnej podoby evolučného algoritmu, zameranej na nájdenie jedného riešenia. Postup by mohol vyzerať napríklad nasledovne:

1. Algoritmus sa použije na nájdenie riešenia, ktoré sa následne zaradí do prázdnej množiny doposiaľ nájdených riešení.
2. Algoritmus sa použije na nájdenie ďalšieho riešenia, pričom sa neberie do úvahy, ktoré riešenia už sú známe.
3. Ak nájdené riešenie je už zaradené v množine doposiaľ nájdených riešení, tak sa ignoruje. Ak sa jedná o doteraz neznáme riešenie, tak je vložené do množiny doposiaľ nájdených riešení.
4. Ak kardinalita množiny doposiaľ nájdených riešení je menšia než je požadovaná, tak sa pokračuje návratom k bodu 2. V opačnom prípade hľadanie končí.

Úspešnosť takéhoto iteračného postupu hľadania viacerých riešení stojí a padá na bode 3 – podľa toho aká je frekvencia nachádzania nových riešení

oproti frekvencii objavovania riešení už známych. Ak uvedeným iteračným spôsobom je potrebné nájsť n riešení, tak je potrebné realizovať

$$nR \tag{1.1}$$

iterácií (cyklov) hľadania, kde R reprezentuje faktor redundancie.

Predpokladajme, že existuje n riešení, pričom nájdenie každého z nich je rovnako pravdepodobné. V tomto prípade na nájdenie prvého riešenia postačuje jedna iterácia. Ak už je známych k riešení, potom pravdepodobnosť objavenia nového riešenia je $(n-k)/n$ a teda pre jeho nájdenie je potrebné vykonať $n/(n-k)$ iterácií. Celkový počet iterácií potrebný pre nájdenie všetkých n riešení je

$$nR = 1 + \frac{n}{n-1} + \frac{n}{n-2} + \ldots + \frac{n}{1} = n \sum_{j=1}^{n} \frac{1}{j} \tag{1.2}$$

pričom sumačný člen je vyjadrením faktora redundancie.

Toto vedie na prijateľnú redundanciu pre uvedené iteračné hľadanie viacerých riešení. Ak však nájdenie rôznych riešení nie je rovnako pravdepodobné, tak redundancia narastá do v praxi neprijateľných hodnôt. Preto sa používajú varianty evolučných algoritmov, ktoré sú špecializované na hľadanie viacerých riešení (angl: niche methods). Tieto varianty je možné rozdeliť do dvoch základných tried:

- paralelné hľadanie – všetky riešenia sa hľadajú naraz počas jednej realizácie algoritmu,
- sekvenčné hľadanie – hľadanie riešení je iteračné, keď v každej iterácii sa hľadá iba jedno (doteraz neznáme) riešenie.

1.1 Paralelné hľadanie viacerých riešení

Vykonáva sa iba jedna iterácia hľadania, pričom sa hľadajú viaceré riešenia súčasne. Postup je založený na myšlienke rozbitia jednoliatosti populácie jedincov tak, aby rozličné časti populácie vypĺňali rozličné "životné priestory" v priestore prehľadávania.

Aj keď je možné použiť prístupy špecializované pre tento typ problémov, napríklad multinacionálne algoritmy [43], paralelné prístupy sú typicky reprezentované "klasickejšími" metódami, použiteľnými aj pre udržiavanie rôznorodosti populácie a tým zabraňujúce jej konvergencii do oblasti okolo jedného riešenia [26]. No nie všetky takéto metódy sú použiteľné, iba tie,

ktoré dokážu dlhodobo udržiavať niekoľko subpopulácií v rámci populácie, konvergujúcich každá k inému riešeniu.

Používané prístupy možno rozdeliť na základe toho, ktorú fázu evolučného cyklu štandardného evolučného algoritmu modifikujú. Toto kritérium vedie na dve hlavné triedy, zamerané na

- selekciu jedincov do úlohy rodičov,
- vytváranie novej generácie.

Samozrejme, je možné použiť aj metódy patriace súčasne do oboch tried, snažiace sa spojiť výhody oboch alternatív. Takou metódou je napríklad seceder operátor prezentovaný v ďalšom texte alebo metóda MNC (multi-niche crowding) prezentovaná v [44].

1.1.1 Metódy orientované na výber rodičov

Cieľom je ovplyvňovať proces vytvárania skupiny rodičov takým spôsobom, aby sa vytvoril čo najlepší predpoklad pre zabránenie konvergencie populácie do úzko ohraničeného podpriestoru. Ovplyvňovať je možné:

- vhodnosť jedincov, na základe ktorej sú jedince selektované,
- výber jedincov do úlohy rodičov a ich párovanie pre aplikáciu binárnych a viacárnych genetických operátorov.

Druhá skupina obsahuje metódy, založené na predstave podobnosti a odlišnosti jedincov. Podľa tejto predstavy v populácii existujú jedince, ktoré sú si navzájom podobné ako aj jedince, ktoré sú navzájom od seba odlišné. Aby sa zabránilo konvergencii jedincov do malého podpriestoru (a tým nájdeniu iba jedného riešenia), populácia by mala byť vytváraná odlišnými jedincami, resp. by v nej mal byť dostatočne veľký počet odlišných jedincov[1].

Prítomnosť odlišných jedincov v populácii je možné podporovať rozhodnutiami, majúcimi za následok výber jedincov berúci do úvahy podobnosť a odlišnosť jedincov. Napríklad keď budú realizované takým spôsobom, aby počet odlišných jedincov ostával v populácii čo najväčší. Pre výber rodičov to znamená identifikovať odlišných jedincov v populácii a preferovať ich do úlohy rodičov. Pre párovanie to znamená kombinovať podobných rodičov.

Väčšiu popularitu si však získali metódy, patriace do prvej skupiny, ktoré vlastne reprezentujú penalizačný prístup. Cieľom tohto prístupu je potláčať

[1] Podobnosť alebo odlišnosť jedincov je možné určovať ako ich vzdialenosť alebo ako ich príslušnosť k rovnakým či navzájom rôznym "druhom" (vyjadrenú napríklad explicitne príznakom druhu).

zhlukovanie jedincov a naopak podporovať rovnomernejšie pokrytie priestoru jedincami populácie. Spočíva na predstave, že zdroje, ktoré majú k dispozícii jedince obývajúce nejaký podpriestor, sú obmedzené. Čím viac jedincov sa nahromadí v nejakom podpriestore, tým menej zdrojov pripadá na jedného jedinca. Výsledkom je nemožnosť plne uspokojovať potreby jedincov v danom podpriestore a následkom toho dochádza iba k obmedzenému prideľovaniu zdrojov.

Analogicky k tejto predstave úlohu zdrojov hrá vhodnosť. Obmedzené prideľovanie zdrojov je reprezentované situáciou, keď vhodnosť jedincov je znižovaná. Jedince sú penalizované za zhlukovanie sa v malom podpriestore. Čím dochádza k väčšej koncentrácii jedincov v nejakom podpriestore, tým viac je vhodnosť týchto jedincov penalizovaná.

Ak teda časť populácie skonverguje do malého podpriestoru, jedincom nachádzajúcim sa v tomto podpriestore je ich vhodnosť dočasne znížená (počas selekcie skupiny rodičov). Vďaka tomu jedince z tohto podpriestoru majú menšiu šancu stať sa rodičmi a tým sa zároveň znižuje pravdepodobnosť, že aj novo generovaní potomkovia budú patriť do daného podpriestoru. Následkom toho v nasledujúcej generácii možno očakávať menej jedincov patriacich do daného podpriestoru ako v prípade, ak by jedince neboli penalizované za svoju vzájomnú blízkosť.

Najznámejším reprezentantom tohto prístupu je metóda *zdieľania* (angl: sharing) [7]. Pri tejto metóde je vhodnosť každého jedinca v populácii v každej generácii premapovaná na novú hodnotu podľa vzťahu

$$\Phi'(a_i(t)) = \frac{\Phi(a_i(t))}{\sum_{j=1}^{\mu(t)} g\left(\, d(a_i(t), a_j(t))\, \right)} \tag{1.3}$$

kde Φ je vhodnosť jedinca, g je zdieľacia funkcia, d je vzdialenosť a μ je veľkosť populácie.

Zdieľacia funkcia je funkciou vzdialenosti medzi dvomi jedincami populácie. Nadobúda hodnotu 1.0 v prípade, že jedince sú identické (teda zdieľajú ten istý bod priestoru a ich vzdialenosť je minimálna) a hodnotu 0.0 ak vzdialenosť prekročí nejaký zadaný prah. Tento prah určuje veľkosť oblasti v ktorej sa uvažuje penalizácia – jedinec nie je ovplyvňovaný tými jedincami, od ktorých je vzdialenejší ako tento zadaný prah. Najčastejšie používaná zdieľacia funkcia má tvar

$$g(d) = \begin{cases} 1 - \left(\frac{d}{r}\right)^{\alpha}, & d \leq r \\ 0, & d > r \end{cases} \tag{1.4}$$

kde r reprezentuje zadný prah označovaný aj ako polomer zdieľania (angl: niche radius). Parameter α je využiteľný k regulovaniu tvaru funkcie, avšak

často je použité $\alpha = 1.0$ aby sa nezvyšoval zbytočne počet parametrov, ktoré je potrebné nastaviť. Inou alternatívou je funkcia stupňovitého tvaru

$$g(d) = \begin{cases} 1, & d \leq r \\ 0, & d > r \end{cases} \tag{1.5}$$

V prípade prvej zdieľacej funkcie je jedinec penalizovaný tým viac, čím je viac jedincov v jeho blízkom okolí alebo čím sú tieto jedince k nemu bližšie. V druhom prípade záleží iba na počte blízkych jedincov ale nie na ich blízkosti (iba na tom či sú bližšie alebo ďalej ako je polomer zdieľania).

V úlohe vzdialenosti je možné použiť Hammingovu vzdialenosť priamo pre reprezentácie jedincov (danú ako počet stavebných blokov štruktúr jedincov majúcich rôzne hodnoty v jedincoch) alebo Euklidovu vzdialenosť pre dekódované jedince (resp. nejakú vzdialenosť založenú na problémovo špecifických znalostiach) [25].

Problematickou časťou metódy je nastavenie vhodnej hodnoty polomeru zdieľania. Pri príliš malej hodnote tohto polomeru penalizácia bude pôsobiť iba v malej oblasti a následkom toho sa nezabráni konvergencii jedincov do oblasti jedného z riešení. Naopak, príliš veľká hodnota polomeru môže mať za následok, že jedince nielen že nebudú konvergovať do oblasti určitého riešenia, ale budú vytlačené aj z okolia susedných riešení.

Neexistuje všeobecne použiteľná hodnota polomeru zdieľania, ktorá by bola použiteľná pre každý individuálny prípad. Návrh horného odhadu vychádza z predstavy priestoru prehľadávania ako k-rozmernej jednotkovej hyperkocky – priestoru s k dimenziami, pričom dĺžka každej z nich je jednotková. Nech v takejto hyperkocke je umiestnených $n = m^k$ bodov reprezentujúcich hľadané riešenia a každý z nich je centrom hypersféry s polomerom r. Ak majú tieto hypersféry čo najlepšie vyplniť vnútro priestoru bez toho aby sa navzájom prekrývali, potom riešenia musia byť v priestore rozmiestnené rovnomerne. Pri takomto rozmiestnení riešenia budú v smere každej osi rozdelené do m radov. Pre hodnotu polomeru potom vychádza

$$r = \frac{1}{2\sqrt[k]{m^k}} \tag{1.6}$$

Pri praktickom použití problému s n riešeniami (n skôr nie je ako je presnou mocninou) sa môže použiť normalizácia priestoru prehľadávania na jednotkovú hyperkocku a hodnota m sa určí tak, aby platilo

$$m - 1 < \sqrt[k]{n} \leq m \tag{1.7}$$

Jednotlivé riešenia sú však málokedy rozmiestnené rovnomerne. Následkom toho budú existovať také riešenia, ktorých ideálna hodnota polomeru je menšia ako hodnota vypočítaná. Preto takto určenú hodnotu možno použiť iba ako horný odhad hodnoty polomeru. Snaha zohľadniť nerovnomerné umiestnenie riešení viedla k niekoľkým modifikáciám, využívajúcim premenlivú veľkosť polomeru zdieľania. Príkladom môže byť prístup umožňujúci adaptáciu penalizačných oblastí [12].

Iný spôsob použitia penalizácie je rozdeliť celú populáciu na explicitné zhluky jedincov a pri určovaní penalizácie uvažovať vždy iba jedince určitého zhluku. Keďže zhluk je explicitne daný, penalizácia môže byť variabilnejšia. Môže byť založená na:

- vzdialenosti jedincov (či už navzájom alebo od centra zhluku),
- počte jedincov prislúchajúcich k danému zhluku,
- selekcii, keď iba niekoľko jedincov (zvyčajne tých najlepších) z každého zhluku môže byť vyberaných do funkcie rodičov [36].

Uvedené alternatívy je možné kombinovať – príkladom je penalizácia založená súčasne na vzdialenosti od centra zhluku ako aj na počte jedincov v danom zhluku, realizovaná podľa vzťahu

$$\Phi'(a_i(t)) = \frac{\Phi(a_i(t))}{n_j \left(1 - \left(\frac{d_{ij}}{2r}\right)^{\alpha}\right)} \tag{1.8}$$

kde n_j je počet jedincov patriacich do zhluku j a d_{ij} je vzdialenosť i-teho jedinca od centra zhluku j (predpokladá sa, že jedinec a_i je členom zhluku j). Ak nejaký jedinec je sám v príslušnom zhluku, tak jeho vhodnosť nie je penalizovaná, pretože zároveň reprezentuje aj centrum svojho zhluku. Čím viac jedincov patrí do daného zhluku alebo čím bližšie leží jedinec k centru zhluku, tým je jeho penalizácia väčšia.

Pre určovanie zhlukov je možné použiť niektorý z existujúcich zhlukovacích algoritmov. Pretože však realizácia takéhoto zhlukovania môže predstavovať veľkú výpočtovú záťaž, je možné používať aj jednoduchšie techniky na iba orientačné určenie zhlukov. Jedna z nich je *dynamické zhlukovanie* (angl: dynamic niching):

1. Populácia jedincov sa zotriedi podľa vhodnosti od lepších k horším hodnotám.

2. Postupne sa prechádza populáciou a definujú sa centrá zhlukov. Ak jedinec je dostatočne blízko centra niektorého zo zhlukov, tak sa prechádza na ďalšieho jedinca. Inak sa daný jedinec stáva centrom nového zhluku.

3. Každý jedinec je zaradený do toho zhluku, k centru ktorého sa nachádza najbližšie.

Tento postup je možné modifikovať tak, že sa vytvára iba maximálne určitý počet zhlukov alebo jedinec je zaradený do zhluku iba ak jeho vzdialenosť od centra daného zhluku je menšia ako nejaký zadaný prah.

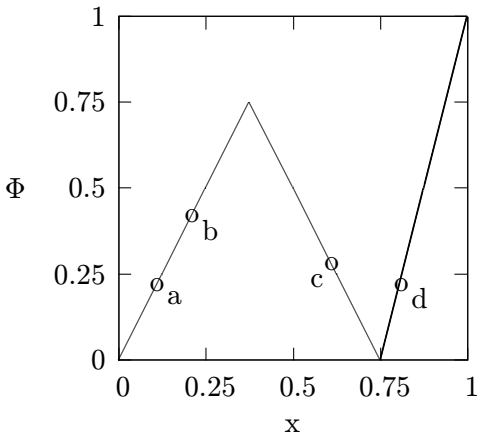
Odbúrať tvorbu zhlukov nanovo v každom evolučnom cykle sa snaží zhlukovacia metóda *DNC* (angl: dynamic niche clustering) prezentovaná v [11], založená na explicitnej reprezentácii množiny penalizačných oblastí s rôznou hodnotou polomeru. Na začiatku sa vytvorí sada oblastí – každému jedincovi bude zodpovedať jedna oblasť, pričom budú mať rovnaké polomery a jedince budú predstavovať stredy týchto oblastí. V každom evolučnom cykle sa vykoná:

- zmení sa poloha stredu každej oblasti tak, aby reflektovala rozloženie jedincov patriacich do danej oblasti,
- ak nejaká oblasť neobsahuje žiadneho jedinca, tak sa zruší; ak nejaký jedinec nie je členom žiadnej oblasti, tak sa vytvorí nová oblasť,
- ak stredy dvoch oblastí sú bližšie ako zadaný prah, tak sa oblasti zlúčia,
- ak sa oblasti prekrývajú ale ich stredy sú od seba ďalej ako je daný prah, tak sú separované (ich polomery sú zmenšené tak, aby k prekrývaniu nedochádzalo).

Následne sa o každom jedincovi rozhodne, do ktorej oblasti patrí, a každý jedinec je penalizovaný vzhľadom na obsadenosť jeho domovskej oblasti.

Uvedené metódy sa pri identifikácii podobnosti jedincov spoliehali na vzdialenosť medzi jedincami – menšia vzdialenosť indikovala väčšiu podobnosť než vzdialenosť väčšia. Toto však nemusí platiť vždy, pretože z hľadiska hľadania viacerých riešení sú dva jedince, ktoré identifikujú rovnaké riešenie ("smerujú" k rovnakému riešeniu), podobnejšie ako jedince, identifikujúce rôzne riešenia – a to aj v prípade, ak vzdialenosť medzi jedincami z rôznych riešení je menšia ako vzdialenosť medzi jedincami patriacimi k tomu istému riešeniu. Táto situácia je ilustrovaná na obr. 1.1.

V danej situácii je vzdialenosť jedinca c k jedincovi d menšia ako k jedincovi b a teda jedinec c by mal byť penalizovaný skôr kvôli jedincovi d



Obr. 1.1: Ilustrácia vzťahu vzdialenosti a podobnosti jedincov.

než kvôli jedincovi b. Avšak z hľadiska rozmiestnenia jedincov sú pomery obrátené – keďže jedince c a d identifikujú rôzne riešenia, ich vzájomná penalizácia nie je potrebná. Naopak, je možné uvažovať o vzájomnej penalizácii medzi jedincami b a c, pretože identifikujú to isté riešenie.

Z tohto dôvodu prístup podľa [21] uvažuje pri určovaní podobnosti nielen vzdialenosť jedincov ale aj ich vzájomnú "orientáciu". Orientácia je definovaná smerom vzostupu hodnôt vhodnosti v okolí jedinca. Na obrázku majú jedince a, b a d orientáciu doprava a jedinec c je zase orientovaný doľava. Vzájomná orientácia dvoch jedincov potom môže byť "od seba" (c a d), "k sebe" (a a c alebo b a c) alebo "za sebou" (a a b).

Na základe vzdialenosti a vzájomnej orientácie sú dva jedince považované za podobné iba v prípade, ak ich vzdialenosť je menšia ako nejaká stanovená hodnota r a ich vzájomná orientácie je k sebe alebo za sebou. Ak sú dva jedince orientované od seba, potom sú považované za nepodobné bez ohľadu na ich vzájomnú vzdialenosť.

1.1.2 Metódy orientované na tvorbu novej generácie

Cieľom je ovplyvňovať proces formovania novej generácie takým spôsobom, aby sa bránilo jedincom vytváranej novej generácie konvergovať smerom do úzko ohraničeného podpriestoru. Ovplyvňovať je možné:

- výber jedincov v populácii, ktoré majú byť nahradené novými jedincami,

- štruktúru jedincov vkladaných do aktuálnej populácie.

Pri výbere jedincov, ktoré majú byť nahradené, sa vychádza z predstavy "druhov" ako subpopulácií, keď každá subpopulácia je určená na objavenie iného riešenia. Aby druhy v populácii prežívali, tak potom jedinec nejakého druhu by mal nahradiť jedinca svojho druhu – v opačnom prípade by mohlo dochádzať k redukcii počtu druhov.

Pretože často nie je druh explicitne identifikovaný, vyvstáva problém ako určiť, či dva jedince patria do toho istého druhu alebo sú členmi rôznych druhov. Keďže skúsenosť hovorí, že vnútrodruhová podobnosť je väčšia ako podobnosť medzidruhová, tak príslušnosť k rovnakému druhu je možné nahradiť podobnosťou jedincov. Ak sú jedince podobné, tak sú považované za príslušníkov toho istého druhu – a čím je táto podobnosť väčšia, tým je ich druhová zhodnosť istejšia.

Noví potomkovia nebudú súťažiť so všetkými pôvodnými jedincami ale ich súťaž sa zameriava iba na jedince, ktoré sú s nimi podobné (s úmyslom nahrádzať iba jedincov patriacich do rovnakej subpopulácie ako tieto novo generované jedince) [23].

Jednoduchým zosobnením tohto postupu by bolo pre každého potomka zistiť, ktorý z pôvodných jedincov, tvoriacich populáciu, je mu najviac podobný. S týmto by potom súťažil o svoje zaradenie do populácie. V najjednoduchšom prípade by toho jedinca jednoducho v populácii nahradil, avšak je možný aj deterministický alebo stochastický súboj [28] založený na ich vhodnostiach[2]. Z dôvodu zníženia výpočtovej náročnosti je možné nového jedinca porovnávať iba so vzorkou populácie namiesto s celou populáciou. Túto vzorku je možné vybrať náhodne (s prihliadnutím alebo bez prihliadnutia k vhodnosti jedincov) alebo deterministicky (napr. najhoršie jedince). Tento prístup je reprezentovaný metódou zvanou *crowding*[3] (angl: crowding).

Ešte väčšie zjednodušenie prináša *deterministický crowding* (angl: deterministic crowding) založený na predpoklade, že novo vygenerované jedince sú najpodobnejšie svojim rodičom. A tak buď priamo nahradia svojich rodičov v populácii alebo s nimi súťažia v deterministických alebo stochastických súbojoch.

Keďže pri krížení z dvoch rodičov vznikli dvaja potomkovia, existujú dve možnosti organizácie súťaže:

[2]Pri deterministickom súboji by vyhral jedinec s lepšou vhodnosťou, pri stochastickom by na vhodnosti jedincov záviseli pravdepodobnosti ich výberu.

[3]Ak prezentované metódy nemajú ustálené slovenské pomenovanie ani "rozumný" preklad, tak je použitý pôvodný anglický názov.

rodič 1	—	potomok 1	rodič 1	—	potomok 2
rodič 2	—	potomok 2	rodič 2	—	potomok 1

V prípade väčšieho počtu rodičov a potomkov sa počet možných usporiadaní súťaže zvyšuje. Pri viacerých možnostiach sa použije to usporiadanie, ktoré minimalizuje priemernú vzdialenosť "rodič-potomok" v rámci súťaže.

Prístupy zamerané na kontrolu štruktúry jedincov, ktoré sú vkladané do populácie, reflektujú potrebu vkladať do populácie tie stavebné prvky, ktoré v nej chýbajú alebo ich výskyt je zriedkavý, pretože boli vytlačené konvergenciou populácie. V opačnom prípade, teda pri vkladaní takého materiálu do populácie, ktorý sa v nej hojne vyskytuje, dochádza k strate rôznorodosti v populácii s následkom straty schopnosti populácie objavovať viaceré riešenia súčasne.

Príkladom tohto prístupu je *operátor jedinečnosti* (angl: uniqueness operator). Pred tým, ako je nejaký novo vytvorený jedinec vložený do populácie, je tento jedinec porovnávaný s tými jedincami, ktoré sa už nachádzajú v populácii. Ak sa zistí, že v populácii už existuje rovnaký jedinec, tak nový jedinec sa do populácie nevkladá. Buď sa priamo zahodí alebo sa urobí pokus zmutovať tohto jedinca tak, aby sa líšil od jedincov v populácii. Táto základná schéma môže byť modifikovaná niekoľkými spôsobmi:

- aby bol jedinec vložený do populácie, tak vzdialenosť[4] medzi ním a každým z ostatných jedincov musí byť väčšia ako zadaný prah,
- pokusov o mutáciu nového jedinca možno vykonať v prípade opakovaného neúspechu viac. Ak ani po vyčerpaní všetkých možností sa nepodarilo jedinca upraviť vhodným spôsobom, tak je zahodený.

Trochu iným prístupom je osobitné udržiavanie variability populácie, ktoré nie je naviazané priamo na vkladanie nových jedincov, ale zvyčajne tvorí až nasledujúcu fázu po ich vložení. Vychádza sa z predpokladu, že pri dodatočnom zabezpečení dostatočnej variability stavebných prvkov je algoritmus schopný vytvárať jedincov ľubovoľnej štruktúry.

Do tejto kategórie patrí *permanentné vkladanie* (angl: permanent load) snažiace sa vkladať do populácie tie stavebné prvky, ktoré v nej chýbajú. Realizácia tejto metódy je veľmi jednoduchá – v každom evolučnom cykle sa niekoľko jedincov z populácie nahradí náhodne generovanými jedincami.

Samotné náhodné generovanie môže zohľadňovať aktuálnu situáciu v populácii alebo ju môže ignorovať. V prvom prípade sa určuje distribúcia stavebných blokov, z ktorých sú jedince skladané, a na jej základe sa rozhoduje

[4]Zvyčajne sa používa Hammingova vzdialenosť.

čo v populácii chýba, respektíve čoho je nedostatok. Následne novo generované jedince sú vytvárané tak, že informáciu o distribúcii jednotlivých stavebných prvkov použijú pre určenie pravdepodobnosti výberu týchto prvkov. Napríklad, ak pri použití binárnej reprezentácie sa zistí, že aktuálne sa v populácii μ jedincov na nejakej pozícii vyskytuje N_0 hodnôt 0, tak táto hodnota bude vkladaná do generovaných jedincov na danú pozíciu s pravdepodobnosťou

$$\frac{\mu - N_0}{\mu} \tag{1.9}$$

zaručujúcou tým väčšiu šancu pre danú hodnotu, čím je zastúpenie tejto hodnoty v populácii menšie.

Pri ignorovaní aktuálnej situácie sa jednotlivé hodnoty vyberajú do generovaných jedincov s rovnakou pravdepodobnosťou, nezávisle od frekvencie ich výskytu v populácii.

1.2 Sekvenčné hľadanie viacerých riešení

Pri jednej iterácii (realizácii) evolučného algoritmu sa hľadá iba jedno riešenie. Po jeho nájdení sa realizuje ďalšia iterácia s cieľom nájsť nové doposiaľ neobjavené riešenie. A to sa opakuje buď pre všetky riešenia (ak je známy ich celkový počet) alebo iba pre žiadaný počet riešení.

Ukážkou tohto prístupu je sekvenčná metóda prezentovaná v [1] (angl: sequential niche). Vychádza z myšlienky, že po nájdení nejakého riešenia toto riešenie už nie je zaujímavé a preto je dané riešenie odstránené (je to možné preto, lebo po jeho nájdení jeho poloha už je známa). Pri druhej iterácii hľadania je preto nájdené iné riešenie, ktoré je následne tiež odstránené. Takýmto spôsobom sa pokračuje ďalej a celý proces končí po nájdení žiadaného počtu riešení.

Operácia "odstránenia" riešenia sa deje modifikáciou funkcie vhodnosti. Nech na pozícii nájdeného riešenia sa vyskytuje vrchol (predpokladáme maximalizačnú úlohu). Tento vrchol sa odstráni – nahradí sa depresiou. Takýmto spôsobom sú jedince populácie vďaka selekčnému tlaku, vyvíjanému evolučným algoritmom, tlačené do iných oblastí, čo vedie k skúmaniu zatiaľ nepreskúmaných oblastí priestoru prehľadávania.

Medzi iteráciami evolučného hľadania dochádza teda k zmene vhodnosti. Počas iteračného hľadania sa vhodnosť nemení – používa sa modifikovaná funkcia vhodnosti v tvare

$$\begin{aligned} \Phi_1(a_i) &= \Phi(a_i) \\ \Phi_{k+1}(a_i) &= \Phi_k(a_i) g(a_i, s_k) \end{aligned} \tag{1.10}$$

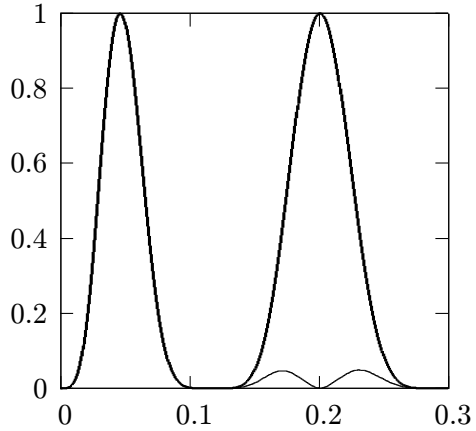
kde Φ_k je vhodnosť použitá v k-tej iterácii pri hľadaní k-teho riešenia a g je modifikačná funkcia, vytvárajúca depresiu v oblasti výskytu riešenia s_k nájdeného v k-tej iterácii. Počas prvotnej iterácie (hľadania prvého riešenia) je teda modifikovaná vhodnosť totožná s pôvodnou funkciou vhodnosti a pre každú ďalšiu iteráciu sa robí modifikácia vhodnosti z predchádzajúcej iterácie s ohľadom na posledné nájdené riešenie.

Modifikačná funkcia je založená na určovaní vzdialenosti[5] medzi dvomi bodmi priestoru prehľadávania, pričom jeden z nich je reprezentovaný jedincom, ktorého vhodnosť má byť modifikovaná, zatiaľ čo druhým bodom je nájdené riešenie, ktoré má byť odstránené. Príkladom takejto modifikačnej funkcie je mocninová funkcia

$$g(a_i, s_k) = \begin{cases} (d(a_i, s_k)/r)^\alpha, & d(a_i, s_k) \leq r \\ 1, & d(a_i, s_k) > r \end{cases} \tag{1.11}$$

kde d označuje vzdialenosť dvoch bodov a r je parameter nazývaný polomer oblasti (angl: niche radius), ohraničujúci oblasť pôsobenia – oblasť, v ktorej dôjde k modifikácii vhodnosti.

Parameter α umožňuje nastaviť vhodný tvar funkcie – v prípade $\alpha = 1$ sa bude jednať o lineárny priebeh, $\alpha < 1$ vytvára konkávny tvar a $\alpha > 1$ zase konvexný tvar. Ilustrácia modifikácie vhodnosti je na obr. 1.2. Hrubšia



Obr. 1.2: Modifikácia vhodnosti potlačením riešenia.

[5]Opäť je možné použiť ako Hammingovu vdialenosť priamo pre reprezentácie jedincov tak aj Euklidovu vzdialenosť pre dekódované jedince.

(horná) čiara znázorňuje pôvodnú (nemodifikovanú) vhodnosť. Táto bola upravená modifikačnou funkciu tak, aby bolo odstránené riešenie $s_1 = 0.2$, pričom bol použitý polomer oblasti $r = 0.1$ a konvexný tvar modifikačnej funkcie – modifikovaná vhodnosť je znázornená tenšou (dolnou) čiarou.

Ako je vidno z obrázka, extrém reprezentujúci dané riešenie bol potlačený, ale jeho odstránenie nie je dokonalé. Došlo k náhrade pôvodného extrému novými lokálnymi extrémami, ktoré sú fiktívnymi riešeniami – neexistujú v pôvodnej nemodifikovanej funkcii. Ich výška závisí od použitej hodnoty parametra α. Čím bude mať väčšiu hodnotu (teda čím bude modifikačná funkcia konvexnejšia), tým bude výška týchto extrémov menšia.

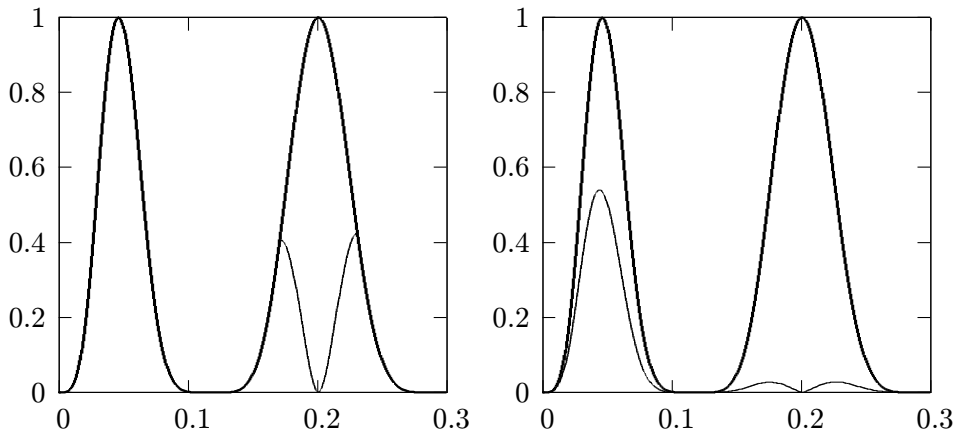
Aj keď sú vytvorené fiktívne extrémy malé, môže dôjsť k situácii, že výsledkom nasledujúceho hľadania ďalšieho riešenia bude práve jeden z týchto fiktívnych extrémov. Preto po nájdení ďalšieho riešenia je potrebná kontrola, či sa jedná o skutočné riešenie alebo iba o fiktívne, ktoré nie je riešením nemodifikovanej vhodnosti. Táto kontrola môže byť realizovaná ako kontrola na:

- výšku nájdeného vrcholu (je však nutné mať doménovo závislé znalosti o minimálnej vhodnosti hľadaných riešení),
- vzdialenosť nájdeného riešenia od už známych riešení (vzdialenosť menšia ako použitý polomer oblasti môže znamenať fiktívne riešenie),
- aplikovanie horolezeckého algoritmu.

Tretí prípad znamená, že horolezecký algoritmus sa použije na zlepšenie kontrolovaného riešenia, pričom sa uvažuje pôvodná nemodifikovaná vhodnosť. Ak kontrolované riešenie je fiktívnym, tak na jeho mieste pôvodná vhodnosť nemá extrém a teda horolezecký algoritmus toto riešenie opraví – zvyčajne na riešenie, ktoré už bolo nájdené skôr a jeho následným odstránením vznikol nájdený fiktívny extrém.

Nastavenie hodnoty polomera oblasti môže mať vplyv na dosiahnuté výsledky. Hodnota $r = 0.1$ bola vhodná pre situáciu znázornenú na obr. 1.2. Situáciu pre príliš malú alebo príliš veľkú hodnotu ilustruje obr. 1.3. Ak je hodnota polomeru oblasti príliš malá (ľavá strana obrázka), vznikajúce fiktívne extrémy sú výraznejšie a je väčšia šanca, že budú algoritmom nájdené namiesto iných skutočných riešení.

Ešte horšia situácia je v prípade, že polomer má príliš veľkú hodnotu (pravá strana obrázka). V tomto prípade dochádza k ovplyvneniu nielen toho vrcholu, ktorý je tvorený práve odstraňovaným riešením, ale aj priľahlých vrcholov. Dochádza nielen k znižovaniu hodnoty susedného extrému (a teda



Obr. 1.3: Modifikácia vhodnosti pri malej (vľavo) a veľkej hodnote (vpravo) polomera oblasti.

klesá pravdepodobnosť nájdenia riešenia zviazaného s týmto extrémom) ale zároveň aj k posunu polohy tohto extrému[6]. V prípade objavenia riešenia, reprezentovaného takto posunutým extrémom, je potrebné korigovať toto nájdené riešenie – pomocou horolezeckého algoritmu nájsť pôvodnú polohu posunutého extrému.

Neexistuje hodnota polomera oblasti, ktorá by bola všeobecne použiteľná, je ju potrebné nastaviť pre každý individuálny prípad. Pre nastavenie horného odhadu je možné použiť rovnaký postup ako pre horný odhad polomera zdieľania (kap. 1.1.1).

1.3 Porovnanie metód pre paralelné hľadanie viacerých riešení

Pre empirické porovnanie popisovaných metód pre paralelné hľadanie viacerých riešení sme použili známu úlohu numerickej optimalizácie – hľadanie maxím multimodálnej funkcie. V úlohe testovacej funkcie bola použitá al-

[6]Situáciu zhoršuje príliš veľká hodnota parametra α – preto pre potlačenie fiktívnych extrémov nie je možné jednoducho zväčšovať hodnotu tohto parametra.

ternatíva funkcie Shekel's Foxholes definovaná vzťahom [24]:

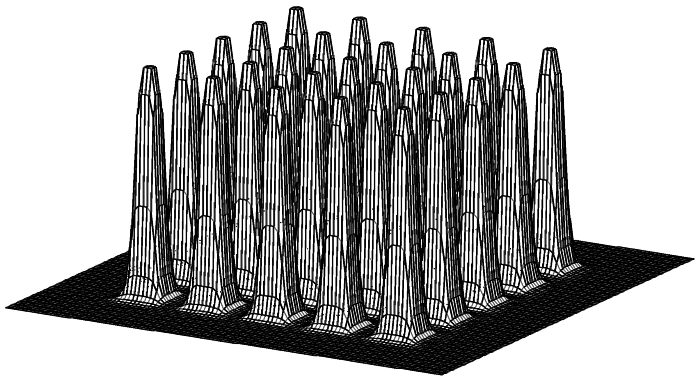
$$f(x,y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1+i+(x-a(i))^6+(y-b(i))^6}} \tag{1.12}$$

kde koeficienty $a(i)$ a $b(i)$ sú definované prostredníctvom vzťahov

$$\begin{array}{rcl} a(i) & = & 16(i\%5 - 2) \\ b(i) & = & 16(\lfloor i/5 \rfloor - 2) \end{array} \tag{1.13}$$

a kde % reprezentuje funkciu modulo a $\lfloor\rfloor$ zase reprezentuje zaokrúhlenie smerom nadol.

Je to dvojdimenzionálna funkcia s dvadsiatimi piatimi vrcholmi rôznej výšky – zobrazená je na obr. 1.4.



Obr. 1.4: Testovacia funkcia Shekel's Foxholes.

Pri realizovaných experimentoch sa obe nezávislé premenné skúmali v intervale $< -64, 64 >$ (vrcholy funkcie sa nachádzajú približne v rámci oblasti $< -40, 40 >$).

Ako testovacie prostredie bol použitý jednoduchý tvar testovacieho algoritmu, ktorý využíval nasledujúce prvky:

- náhodnú inicializáciu populácie (o veľkosti 100 jedincov),
- Grayov binárny kód pre reprezentáciu hodnôt premenných (každá premenná bola reprezentovaná pomocou 17 bitov),
- selekciu 2 rodičov binárnym turnajom bez náhrady,

- tvorbu 2 potomkov dvojbodovým krížením ($p_c = 0.5$) a mutáciou zmeny bitu ($p_m = 1/34$),
- náhradu náhodných jedincov v populácii novo generovanými potomkami (s chránenou elitou 20 najlepších jedincov).

Štruktúra algoritmu umožňovala použitie metód pre paralelné hľadanie viacerých riešení sústreďujúcich sa ako na fázu selekcie, tak aj na fázu nahradzovania aktuálnej generácie novou generáciou.

Chovanie tohto testovacieho algoritmu je zobrazené na obr. 1.5, pričom kvôli získaniu zobrazených dát bolo vykonaných 200 opakovaní a dosiahnuté výsledky boli spriemernené. Sledoval sa priebeh priemernej vhodnosti populácie v závislosti na čase (vľavo hore), počet vrcholov identifikovaných jedincami populácie[7] (vpravo hore) a priemerná vzdialenosť jedincov od najbližšieho maxima[8] (dole).

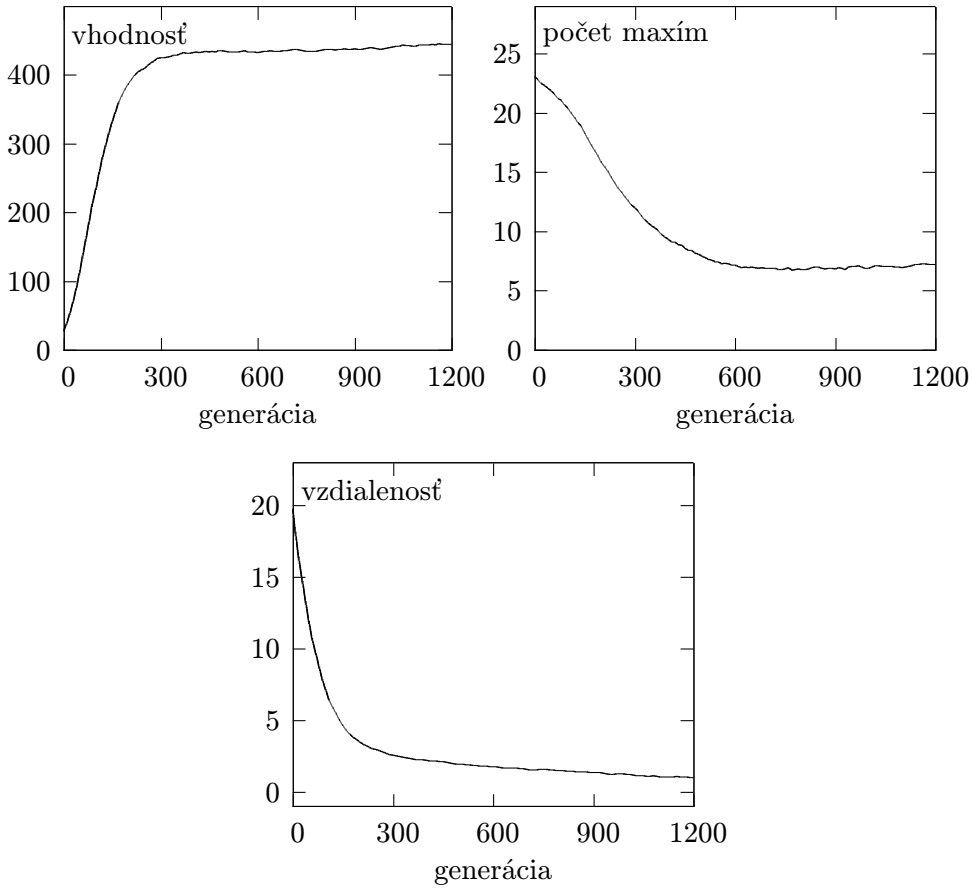
Z grafických priebehov je zrejmé, že algoritmus síce hneď po náhodnej inicializácii identifikuje skoro všetky vrcholy, avšak vďaka konvergencii populácie ich stráca a po uplynutí prechodnej fázy už dokáže identifikovať iba menej než tretinu všetkých vrcholov. V rámci identifikovaných vrcholov je populácia rozdelená na subpopulácie (každá subpopulácia je lokalizovaná v jednom z identifikovaných vrcholov), ktoré naďalej mierne konvergujú, o čom svedčí neustále mierne znižovanie priemernej vzdialenosti jedincov k identifikovaným maximám a zlepšovanie priemernej vhodnosti.

Otázkou je, ako by sa mali uvedené priebehy zmeniť v prípade použitia "dobrej" metódy hľadania viacerých riešení. Vďaka náhodnej inicializácii by štartovacie hodnoty všetkých troch priebehov mali ostať približne rovnaké. Po uplynutí prechodnej fázy by sa mal počet identifikovaných maxím stabilizovať na vyššej hodnote (v ideálnom prípade na hodnote 25) a túto hodnotu udržiavať. Hodnoty priemernej vhodnosti a vzdialenosti by mali závisieť na tom, či metóda povoľuje konvergenciu subpopulácií v jednotlivých vrcholoch alebo nie. Ak je konvergencia povolená, tak priebehy by sa mali blížiť tým na obr. 1.5, v opačnom prípade priemerná vhodnosť bude dosahovať menšie hodnoty a priemerná vzdialenosť zase väčšie hodnoty ako na obrázku. A čím menšia konvergencia, tým väčší rozdiel v dosiahnutých hodnotách.

Do použitého testovacieho algoritmu boli pridávané tieto metódy pre paralelné hľadanie viacerých riešení:

[7] Pre každého jedinca sa realizovalo jeho zlepšovanie pomocou horolezeckého algoritmu (malými krokmi v smere súradných osí) až pokiaľ nebolo dosiahnuté niektoré z maxím funkcie – a príslušný vrchol sa považoval za identifikovaný daným jedincom.

[8] Pre určovanie vzdialenosti bola použitá Manhattan metrika.



Obr. 1.5: Spriemernené výsledky pre testovací algoritmus.

Zdieľanie. Pri penalizácii vhodnosti jedincov bola použitá ako stupňovitá funkcia podľa (1.5) tak aj lineárna zdieľacia funkcia podľa (1.4) s hodnotou $\alpha = 1.0$. Polomer zdieľania bol nastavený na 5 (v prípade vzdialenosti určovanej ako Hammingova vzdialenosť medzi binárnymi reprezentáciami jedincov) a 8 (pri vzdialenosti určovanej ako Euklidova vzdialenosť medzi bodmi definovanými súradnicami reprezentovanými porovnávanými jedincami). V prvom prípade bola hodnota zvolená na základe experimentov[9], v druhom prípade zvolená hodnota reflektovala vedomosti o tvare použitej testovacej funkcie.

[9]Tu ako aj v popise ďalších použitých metód – ak nie je uvedené inak, tak použitá hodnota parametra bola získaná experimentálnym nastavovaním daného parametra.

Dosiahnuté výsledky sú na obr. 1.6 vľavo. Prezentujú použitie iba stupňovitej funkcie, nakoľko obe funkcie viedli k porovnateľným výsledkom. Je zrejmé, že pre danú funkciu je vhodnejšie použitie Euklidovej vzdialenosti, umožňujúcej lepšie udržiavanie identifikovaných maxím.

Dynamické zhlukovanie. Maximálny počet zhlukov bol obmedzený na 25 a jedinec bol priradený k najbližšiemu zhluku iba ak Euklidova vzdialenosť medzi ním a centrom zhluku bola menšia ako 8. Obe hodnoty boli odvodené z vlastností testovacej funkcie.

Boli použité dva prístupy k zohľadneniu spoločného výskytu jedincov v tom istom zhluku. Jeden v rámci zhluku penalizoval vhodnosť jedincov podľa počtu jedincov v danom zhluku, zatiaľ čo druhý v rámci zhluku uvažoval iba určitý počet najlepších jedincov. Keďže zvyšujúci sa počet uvažovaných jedincov mal za následok pokles počtu udržiavaných maxím, uvádzame výsledky pre preferovanie iba najlepšieho jedinca v rámci každého zhluku. Dosiahnuté výsledky sú na obr. 1.6 vpravo.

Crowding. Jedinec, ktorý mal byť vkladaný do populácie, bol porovnávaný s náhodne vybratou vzorkou jedincov. Čím bola táto vzorka väčšia, tým pomalšie stúpala priemerná vhodnosť, tým väčší počet maxím bol udržiavaný a tým pomalšie klesala priemerná vzdialenosť jedincov od najbližšieho maxima. Priebehy zobrazené na obr. 1.7 vľavo reprezentujú situáciu pre dvoch a desiatich jedincov v porovnávacej vzorke.

Deterministický crowding. Pri experimentovaní bol použitý jednak deterministický a jednak stochastický súboj medzi novými jedincami a ich rodičmi. Výsledky sú na obr. 1.7 vpravo. Z grafického zobrazenia je zrejmé, že oba typy súbojov sú z hľadiska identifikácie a udržiavania maxím rovnocenné. Deterministické usporiadanie súbojov má za následok o niečo pomalší vzrast priemernej vhodnosti a pokles priemernej vzdialenosti než stochastické usporiadanie.

Operátor jedinečnosti. Čím viac pokusov o mutáciu vkladaného jedinca je možné vykonať, tým je väčšia šanca že bude dosiahnutá požadovaná rôznorodosť. Experimenty boli robené s možnosťou troch mutácií. Ich výsledky sú na obr. 1.8 vľavo. Ilustrujú fakt, že čím je požadovaná vzdialenosť medzi vkladaným jedincom a jedincami populácie väčšia (bola použitá Hammingova vzdialenosť medzi binárnymi reprezentáciami jedincov), tým je udržiavaných viac maxím avšak nárast priemernej vhodnosti a pokles priemernej

vzdialenosti sú pomalšie (a zároveň sú aj väčšie nároky na výpočtové zdroje, pretože narastá počet realizovaných opravných mutácií).

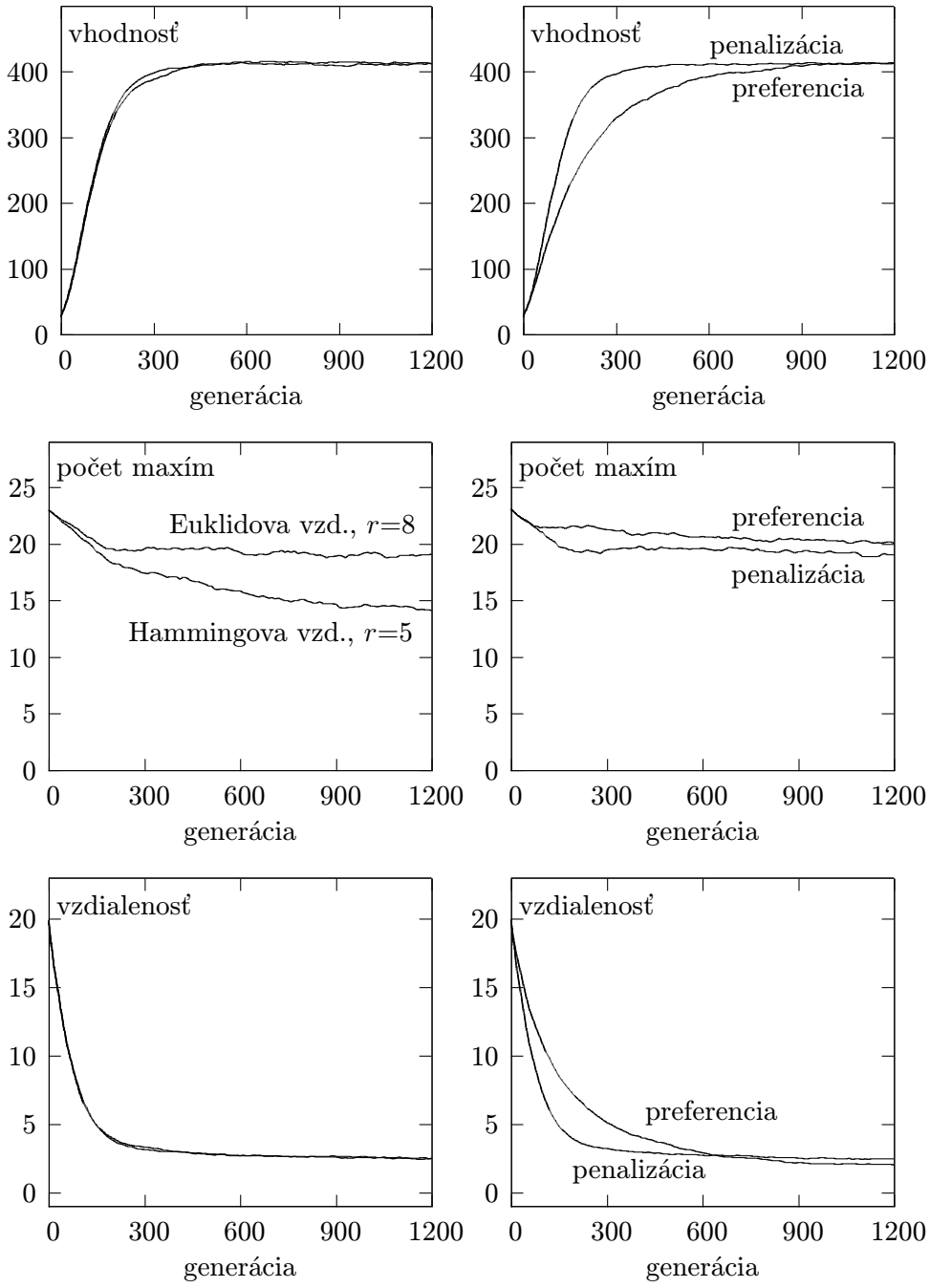
Permanentné vkladanie. Vzhľadom na to, že v každom cykle sú generované a do populácie vkladané iba dva jedince, tak bolo použité vloženie iba jedného náhodne generovaného jedinca v každom cykle. Vzhľadom na počet jedincov v populácii a nízku rýchlosť konvergencie nebol rozdiel medzi generovaním hodnôt 0 a 1 s rovnakou pravdepodobnosťou a medzi generovaním založeným na pravdepodobnostiach, odrážajúcich aktuálnu distribúciu oboch hodnôt v populácii. Grafické priebehy na obr. 1.8 vpravo rozlišujú, či generovaný jedinec nahrádzal najhoršieho jedinca v populácii alebo náhodne vybraného. Deterministická náhrada najhoršieho jedinca vedie síce na priaznivejší nárast priemernej vhodnosti a pokles priemernej vzdialenosti, avšak súčasne aj na udržiavanie menšieho počtu identifikovaných maxím.

Seceder selekcia a náhrada. Metóda založená na podpore jedincov, ktoré sú najviac/najmenej odlišné od ostatných jedincov v populácii. Môže byť použitá ako pri výbere rodičov (najväčšia odlišnosť) tak aj pri vytváraní novej populácie (najmenšia odlišnosť).

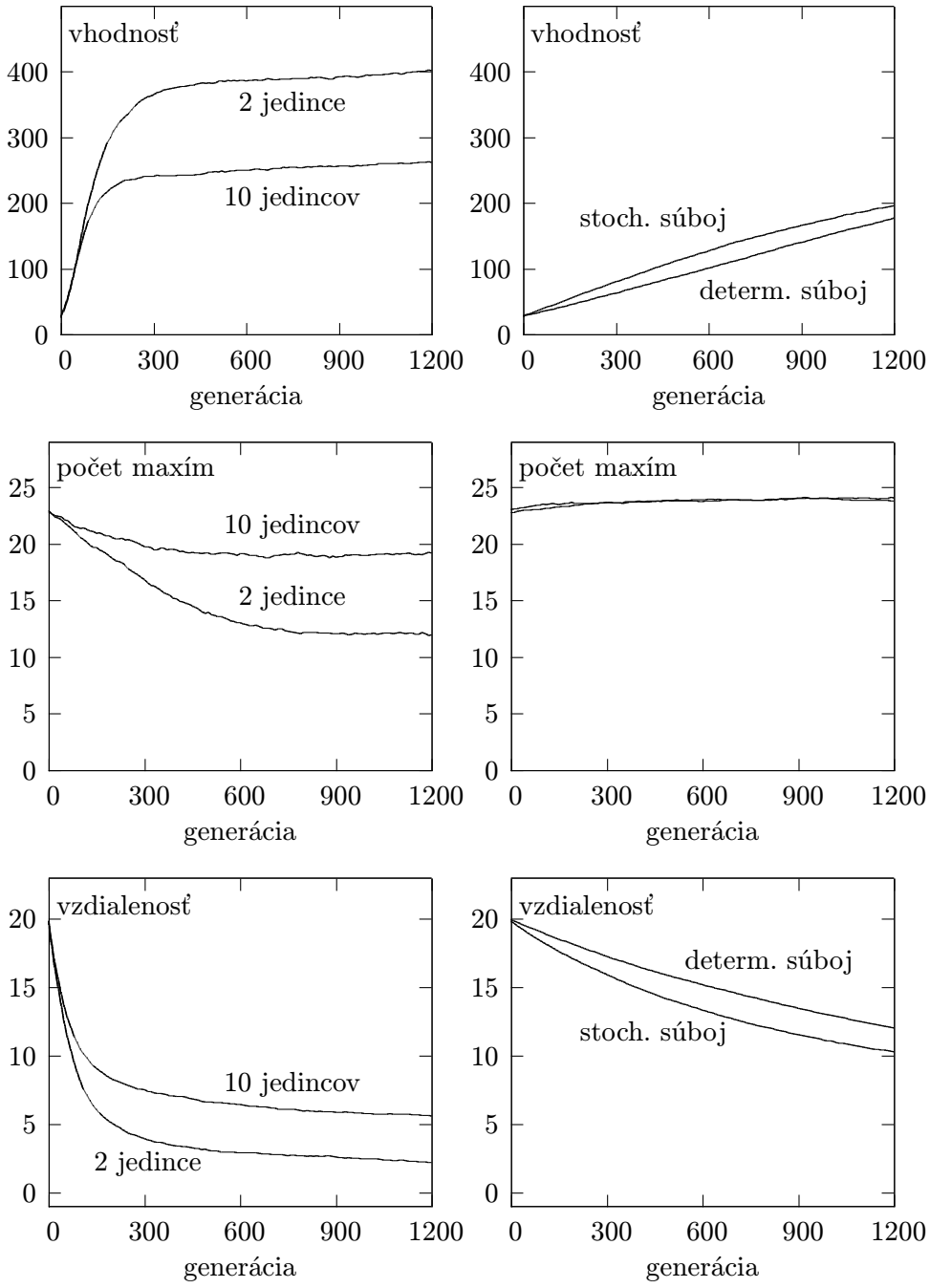
Pre výber každého rodiča sa štandardnou selekčnou metódou vyberie určitý počet jedincov a z nich sa stane ten jedinec rodičom, ktorého vzdialenosť od ostatných vybratých jedincov je najväčšia. Pri náhrade je náhodne vyberaný určitý počet jedincov a z nich je ten nahradený novým jedincom, ktorého vzdialenosť od ostatných vybratých jedincov je najmenšia.

Výsledky dosiahnuté touto metódou sú zobrazené na obr. 1.9. Ľavá strana umožňuje porovnať použitie metódy pri výbere rodičov voči jej použitiu pre tvorbu novej generácie (v oboch prípadoch išlo o výber troch jedincov). Zdá sa, že o niečo úspešnejšie je zaradenie metódy do fázy tvorby novej generácie. Pravá strana obrázka ilustruje súčasné použitie metódy na oboch miestach pre rôzne počty vyberaných jedincov. Je očividné, že kombinácia použitia v selekcii aj v náhrade poskytuje lepšie výsledky ako použitie iba na jednom mieste (porovnanie priebehu pre 3 jedincov voči priebehom v ľavom stĺpci). Súčasne je zrejmé, že čím je výber realizovaný z väčšieho počtu jedincov, tým je udržiavanie maxím lepšie na úkor zhoršenia priemernej vhodnosti a vzdialenosti.

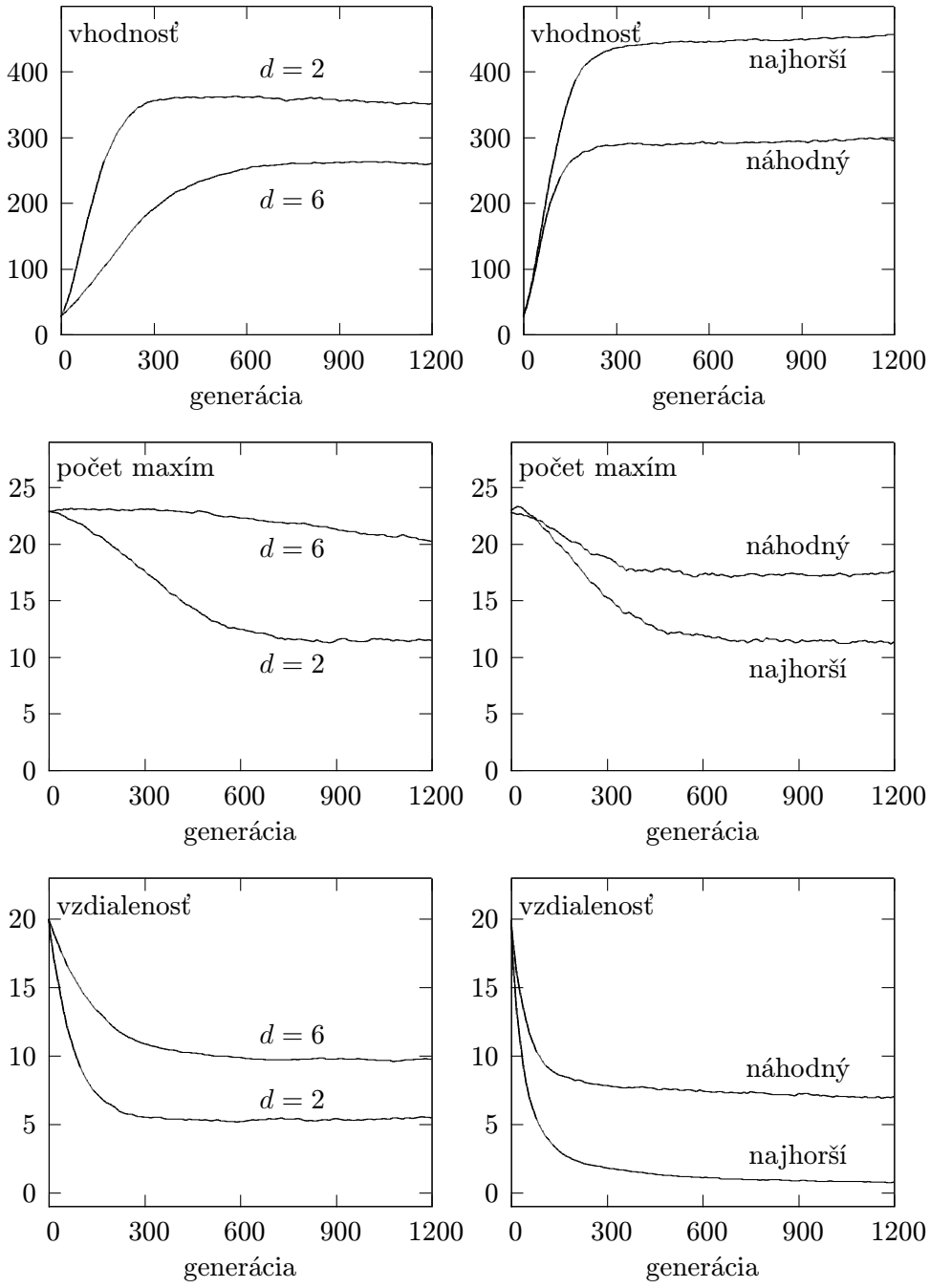
Podľa dosiahnutých výsledkov je možné z hľadiska identifikácie maxím a ich udržiavania počas práce algoritmu rozdeliť testované metódy do troch skupín. Iba jedna metóda bola schopná dosiahnuť vyššiu hodnotu než bola



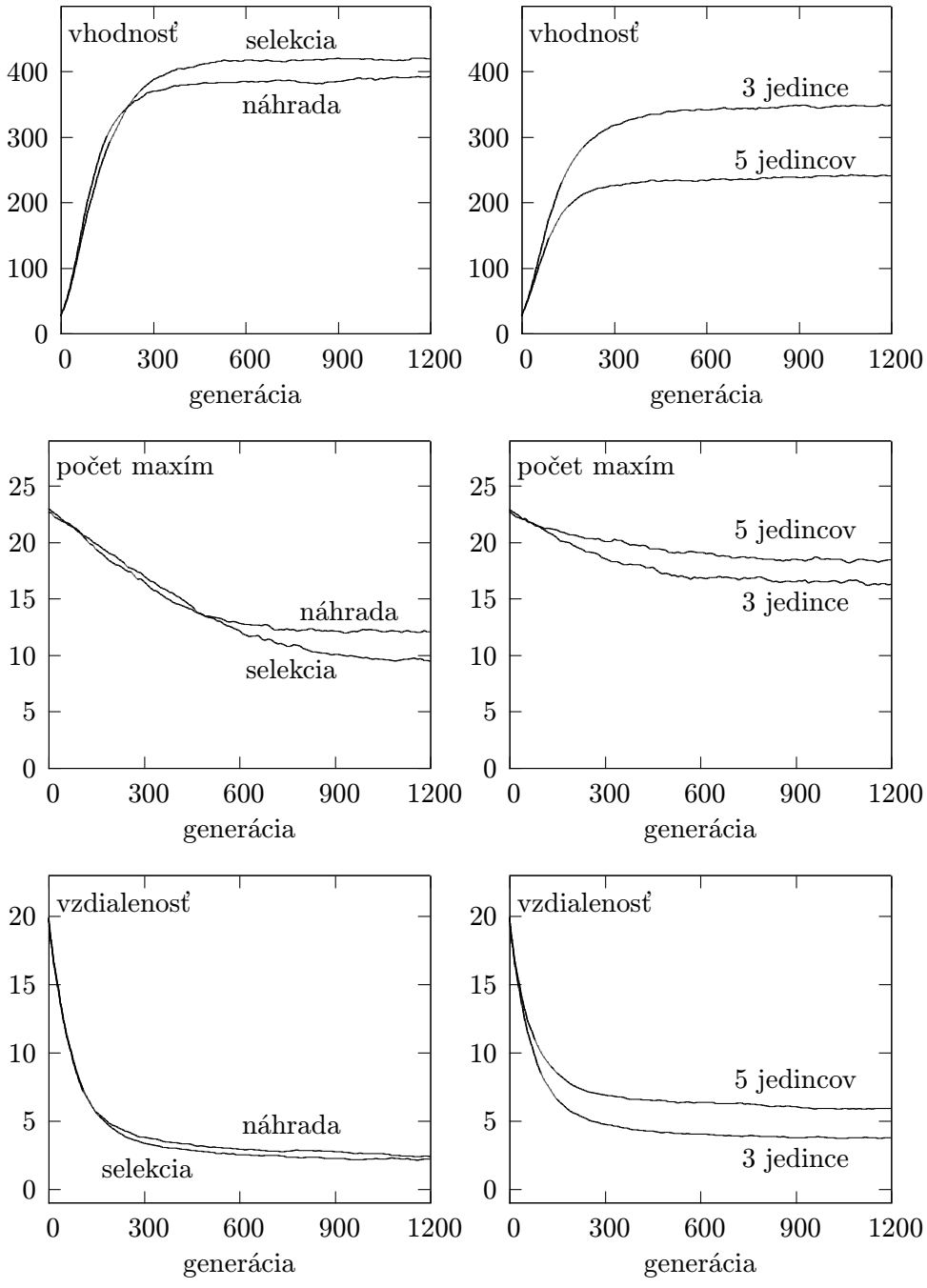
Obr. 1.6: Spriemernené výsledky pre metódu zdieľania (vľavo) a dynamického zhlukovania (vpravo).



Obr. 1.7: Spriemernené výsledky pre metódu crowdingu (vľavo) a deterministického crowdingu (vpravo).



Obr. 1.8: Spriemernené výsledky pre metódu operátora jedinečnosti (vľavo) a permanentného vkladania (vpravo).



Obr. 1.9: Spriemernené výsledky pre seceder operátor použitý v selekcii alebo náhrade (vľavo) a použitý súčasne aj v selekcii aj náhrade (vpravo).

hneď po náhodnej inicializácii populácie na začiatku hľadania – deterministický crowding. Protipólom je seceder operátor použitý iba na jednom mieste, či už v selekcii alebo pri tvorbe novej generácie – kvôli týmto horším výsledkom túto podobu seceder operátora nie je potrebné uvažovať. Ostatné metódy dosiahli v poslednej generácii pomerne blízke hodnoty[10]. Z toho operátor jedinečnosti najprv udržiava iniciálny počet maxím a až okolo 450-tej generácie začína klesať na výslednú hodnotu. U ostatných metód je ten priebeh opačný – najprv nastáva pokles nasledovaný udržiavaním dosiahnutej hodnoty.

Z hľadiska dosahovaných hodnôt[11] priemernej vhodnosti populácie existujú dve základné skupiny. Jedna dosahuje vysokých hodnôt iba o niečo horších ako na obr. 1.5 pri použití základného algoritmu bez podpory hľadania viacerých riešení. Táto skupina je tvorená penalizačnými metódami (zdieľanie a dynamické zhlukovanie). Druhá skupina dosahuje hodnoty iba v strednej tretine rozsahu. Tvoria ju metódy založené na počítaní podobnosti jedincov respektíve na vkladaní náhodných stavebných blokov do populácie. Tieto metódy dosahujú rôznu rýchlosť nábehu – od strmého rýchleho nábehu (permanentné vkladanie) cez pomalší nábeh (operátor jedinečnosti) až po extrémne pomalý nábeh (deterministický crowding). Rovnaké delenie na tie isté skupiny reflektuje aj hľadisko priemernej vzdialenosti jedincov k najbližšiemu maximu.

[10] Pri uvažovaní lepšej z prezentovaných dvoch alternatív – tej, ktorá dosahuje vyššiu hodnotu.

[11] Uvažované sú tie alternatívy, ktoré sú lepšie z hľadiska počtu maxím. Alternatíva identifikujúca menší počet maxím znamená väčšiu konvergenciu populácie a súčasne znamená aj dosiahnutie vyššej priemernej vhodnosti a menšej priemernej vzdialenosti k maximám.

Kapitola 2

Riešenie viackriteriálnych úloh

Pri riešení praktických úloh sa možno často stretnúť s úlohami, ktoré sú charakterizované viacerými kritériami. Pretože nie je vždy možné vyjadriť viacero rozličných účelových funkcií, reprezentujúcich jednotlivé kritériá, pomocou jednej funkcie vhodnosti[1], je nutné uvažovať aj prípady súčasného výskytu viacerých funkcií vhodnosti. V takomto prípade každému jedincovi a_i v populácii namiesto jednej skalárnej hodnoty vhodnosti prislúcha vektor hodnôt vhodnosti

$$[\Phi_1(a_i), \Phi_2(a_i), \ldots, \Phi_n(a_i)] \tag{2.1}$$

kde i-ta hodnota reprezentuje hodnotu i-tej parciálnej funkcie vhodnosti, reprezentujúcej i-tu podmnožinu uvažovaných kritérií.

V prípade použitia iba jednej vhodnosti, pri porovnaní dvoch jedincov s rôznymi vhodnosťami bol za lepšie riešenie považovaný ten jedinec, ktorý mal lepšiu vhodnosť.

V prípade viacerých vhodností je situácia zložitejšia. V tomto prípade je možné porovnanie založiť na pojme dominancie. Pri porovnávaní dvoch vektorov $\vec{u} = [u_1,\ldots,u_n]$ a $\vec{v} = [v_1,\ldots,v_n]$, ak platí vzťah

$$\forall i \in 1, 2, \ldots, n \quad u_i \leq v_i \tag{2.2}$$

pre všetky zodpovedajúce dvojice zložiek a zároveň aj vzťah

$$\exists j \in 1, 2, \ldots, n \quad u_j < v_j \tag{2.3}$$

[1] Obzvlášť v prípade, ak kritériá reprezentujú fundamentálne odlišné aspekty alebo ak sú navzájom konfliktné.

aspoň pre jednu dvojicu zložiek, potom vektor $\vec{v}$ dominuje vektoru $\vec{u}$ (resp. vektor $\vec{u}$ je dominovaný vektorom $\vec{v}$). Formálne sa to znázorňuje ako

$$v \succ u \tag{2.4}$$

To platí v prípade snahy o maximalizáciu všetkých zložiek (v našom prípade parciálnych vhodností). V prípade minimalizácie všetkých vhodností je situácia opačná – vektor $\vec{u}$ dominuje vektoru $\vec{v}$[2]. Súhrnne povedané, jeden vektor dominuje druhému vektoru vtedy, ak je lepší aspoň v jednej zložke a v žiadnej zložke nie je horší.

Pri porovnávaní dvoch jedincov s uvažovaním viacerých vhodností je za lepšieho považovaný ten jedinec, ktorý dominuje druhému. Ak medzi nimi nie je vzťah dominancie, tak sú oba jedince považované za rovnocenné. Jeden z nich môže byť síce lepší podľa jednej alebo niekoľkých vhodností, ale určite je potom horší podľa aspoň jednej z vhodností.

Tie jedince, ktoré nie sú dominované žiadnym iným jedincom, vytvárajú Paretovu množinu. Ak sa pritom uvažuje iba nejaká obmedzená množina jedincov (napríklad jedince aktuálnej populácie), tak hovoríme o lokálnej Paretovej množine. Globálna Paretova množina je tvorená nedominovanými jedincami pri uvažovaní celého priestoru prehľadávania.

Paretova množina v zásade obsahuje dva základné typy jedincov:

- extremálne – jedince veľmi dobré podľa nejakého kritéria (a zvyčajne aj pomerne zlé podľa nejakého iného kritéria),
- kompromisné – jedince, ktoré nie sú výrazne dobré podľa žiadneho kritéria ale rovnako podľa žiadneho kritéria nie sú ani výrazne zlé.

Jedince z Paretovej množiny sú navzájom rovnocenné. Každé jedno riešenie z tejto množiny je optimálne v tom zmysle, že neexistuje žiadne iné riešenie z uvažovaného súboru jedincov, ktoré by bolo lepšie podľa nejakého kritéria a súčasne by nebolo horšie podľa žiadneho iného kritéria.

Zlepšenie niektorej z vhodností môže mať za následok celkové zlepšenie jedinca iba vtedy, ak toto zároveň neprináša degradáciu ostatných parciálnych vhodností. Ak nejaký jedinec nemôže byť takto zlepšený, potom je prvkom Paretovej množiny a je považovaný za optimálneho jedinca.

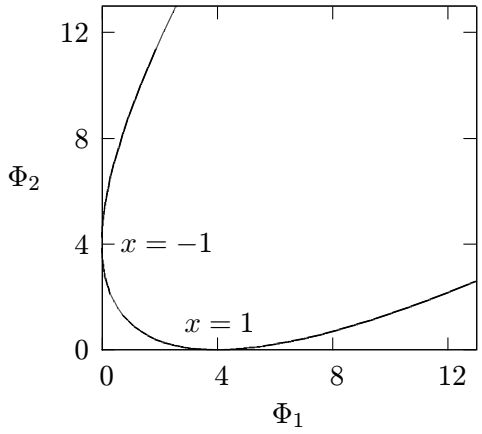
Globálna Paretova množina obsahuje jedince, ktoré sú považované za globálne optimálne a každý z nich je prijateľný ako hľadané riešenie úlohy (každý z nich reprezentuje globálny extrém).

[2]Zložitejšie kombinácie nastávajú, ak niektoré zložky vektora vhodností je potrebné maximalizovať a iné zase minimalizovať.

Pre ilustráciu pojmu Paretovej množiny uvažujme dve funkcie vhodnosti dané v tvare

$$\Phi_1(x) = (x+1)^2 \qquad \Phi_2(x) = (x-1)^2 \tag{2.5}$$

Jedince (z jednorozmerného priestoru prehľadávania definovaného intervalom $< -2.6, 2.6 >$) vytvárajú v dvojrozmernom priestore vhodností krivku podľa obr. 2.1. Ak sa bude uvažovať minimalizácia oboch vhodností, tak Pa-



Obr. 2.1: Ilustrácia Paretovej množiny.

retova množina bude tvorená jedincami $x \in < -1, 1 >$, vytvárajúcimi oblúk medzi bodmi [0,4] a [4,0] (vrátane daných bodov). Všetky jedince $x < -1$ sú dominované jedincom $x = -1$ a všetky jedince $x > 1$ zase jedincom $x = 1$. Žiadny z jedincov $-1 \leq x \leq 1$ nie je dominovaný nijakým iným jedincom. Ak by sa však pre Φ_1 uvažovala maximalizácia a pre Φ_2 opäť minimalizácia, potom by Paretova množina bola tvorená jedincami $1 \leq x$ (všetky jedince $-2.6 < x < 1$ sú dominované jedincom $x = 1$).

Je síce možné v určitom čase (napr. počas jedného evolučného cyklu) používať iba jednu z parciálnych vhodností[3], avšak pre riešenie multikriteriálnych úloh sa v zásade používajú prístupy, zamerané na modifikáciu rôznych častí evolučného algoritmu. Najčastejšie používanými sú tieto:

- nahradenie vektora vhodností jednou syntetickou skalárnou hodnotou, čo umožňuje použiť štandardný tvar evolučného algoritmu,

[3]Výber môže byť náhodný, podľa nejakej preddefinovanej výberovej schémy alebo môže byť založený na progrese dosiahnutom pri použití jednotlivých parciálnych vhodností.

- používanie vektora vhodností s nutnosťou tomu prispôsobiť štandardný tvar evolučného algoritmu.

Uvedené prístupy je možné kombinovať s transformačným prístupom, pri ktorom je vektor parciálnych vhodností premapovaný na iný vektor. Príkladom je [27], kde hodnoty parciálnych vhodností sú transformované na dve kritériá, ktorých sémantika nie je závislá na konkrétnom riešenom probléme.

2.1 Metódy tvorby syntetickej vhodnosti

Cieľom tejto skupiny metód je skalarizácia vektora vhodností do jednej výslednej skalárnej hodnoty. Táto procedúra by mala mať charakter monotónnej transformácie, ktorá by nejakému jedincovi a_i mala priradiť aspoň takú dobrú vhodnosť, ako priradí tým jedincom, ktoré sú dominované daným jedincom a_i – teda po transformácii by žiadny jedinec nemal byť horší ako tie jedince, ktorým dominuje.

V súčasnosti známe metódy je možné rozdeliť do niekoľkých základných skupín:

- agregačné metódy,
- zotrieďovacie metódy,
- metódy založené na dominancii.

2.1.1 Agregácia vhodností

Najjednoduchším spôsobom ako vytvoriť výslednú hodnotu vhodnosti je použiť numerickú kombináciu vhodností, typicky súčet parciálnych vhodností. To je však možné iba vtedy, ak všetky vhodnosti sú rovnakého typu – ak všetky majú byť maximalizované (v tomto prípade je potrebné maximalizovať aj agregovanú vhodnosť) alebo minimalizované (agregovaná vhodnosť má byť minimalizovaná tiež). Ak však niektoré vhodnosti je potrebné maximalizovať a iné minimalizovať, je nutné niektoré z nich transformovať tak, aby všetky parciálne vhodnosti vyžadovali rovnaký spôsob spracovania.

Samotnú agregáciu je možné realizovať ako vážený súčet prostredníctvom vzťahu

$$\Phi'(a_i) = \sum_{j=1}^{n} w_j \Phi_j(a_i) \tag{2.6}$$

kde w_j reprezentuje nezápornú váhu j-tej parciálnej vhodnosti. Tieto váhy však principiálne môžu hrať úlohu dvoch faktorov:

- dôležitosť jednotlivých parciálnych vhodností,
- normalizácia oborov hodnôt parciálnych vhodností.

Čím je kritérium, reprezentované nejakou parciálnou vhodnosťou, dôležitejšie, tým je váha príslušnej parciálnej vhodnosti väčšia. Určenie váh v tomto zmysle však nie je jednoduché, pretože sa môže jednať o kritériá, ktoré sú obtiažne porovnateľné resp. ich porovnanie sa nezvykne vyjadrovať kvantitatívnym spôsobom.

Pri nenormalizovaných parciálnych vhodnostiach sa zase môže stať, že nejaká hodnota môže reprezentovať veľmi dobrú hodnotu pre jednu parciálnu vhodnosť ale pomerne zlú hodnotu pre inú parciálnu vhodnosť. Ak by napríklad obor hodnôt Φ_i bol interval $< 0, 12 >$ a Φ_j by zase nadobúdala hodnoty z $< 10, 100 >$, tak prirodzený dôraz by bol viac na maximalizáciu Φ_j než Φ_i, pretože samotné dosiahnutie maximálnej hodnoty Φ_i nedokáže zabezpečiť také zlepšenie agregovanej vhodnosti než mierne zlepšenie Φ_j. Preto je vhodné realizovať normalizáciu parciálnych vhodností, napríklad podľa

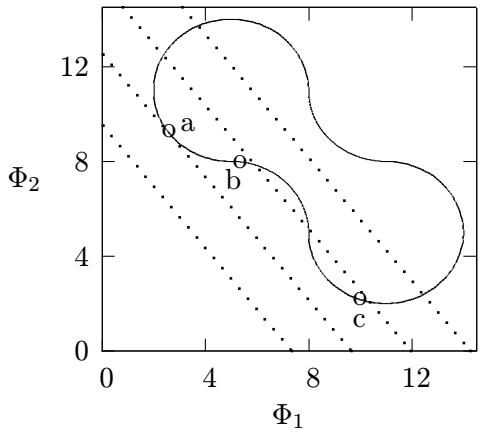
$$\frac{\Phi_j(a_i) - MIN_j}{MAX_j - MIN_j} \tag{2.7}$$

kde MIN_j a MAX_j sú minimálna a maximálna hodnota rozsahu j-tej parciálnej vhodnosti.

Použitie takéhoto normalizačného prepočtu je jednoduché v prípade, že hraničné hodnoty rozsahu sú známe. Horšia situácia je v opačnom prípade – vtedy je ich možné iba odhadovať z hodnôt vhodnosti skúmaných jedincov, a to buď lokálne (v rámci danej generácie) alebo globálne (v rámci aktuálnej aj minulých generácií). Pri globálnom odhade v prípade maximalizácie sa odhad minimálnej hodnoty zvyčajne ustáli v skorých etapách evolučného hľadania, zatiaľ čo odhad maximálnej hodnoty sa zvyšuje aj v neskorších etapách.

Iná nevýhoda použitia agregovanej vhodnosti je ilustrovaná situáciou na obr. 2.2. Obrázok znázorňuje situáciu, keď je potrebné minimalizovať obe vhodnosti, zatiaľ čo jednotlivé riešenia sú zobrazované iba v rámci znázorneného útvaru. Je zrejmé, že všetky tri vyznačené body (ako aj všetky body medzi nimi) reprezentujú riešenia patriace do Paretovej množiny.

Lineárna agregácia vhodností má za následok, že body priestoru vhodností, ktoré majú rovnakú hodnotu agregovanej vhodnosti, vytvárajú v priestore priamku (čím menšia hodnota agregovanej vhodnosti, tým je priamka bližšie počiatku). Obrázok ilustruje štyri takéto priamky pre štyri rôzne hodnoty agregovanej vhodnosti.



Obr. 2.2: Ilustrácia agregácie vhodností.

Nejaká konkrétna kombinácia váh parciálnych vhodností určuje sklon týchto priamok. Je zrejmé, že v situácii na obrázku bod "a" reprezentuje globálny extrém zatiaľ čo bod "c" nie je extrémom. Pri inej kombinácii váh (sklone priamok) by mohla nastať opačná situácia – bod "c" by bol globálnym riešením. Teda dochádza k nerovnosti medzi jednotlivými riešeniami z Paretovej množiny.

Navyše, umiestnenie bodu "b" je také, že pri ľubovoľnej kombinácii váh bude vždy horší ako niektorý z jeho susedov a nikdy nebude globálnym ani lokálnym extrémom. To znamená, že niektoré časti Paretovej množiny úplne strácajú šancu byť nájdené ako riešenie úlohy. Globálne riešenie agregovaného problému je zároveň nedominovaným riešením pôvodného viackriteriálneho problému. Opačne to však neplatí.

Aby sa zvýšila šanca nájdenia viacerých riešení z Paretovej množiny (tých, ktoré vôbec majú šancu byť nájdené), je potrebné použiť rôzne kombinácie váh. Váhy je možné nastavovať napríklad tak, aby platili vzťahy:

$$0 \leq w_1, w_2, \ldots, w_n \leq 1 \qquad w_1 + w_2 + \ldots + w_n = 1 \tag{2.8}$$

Takto nastavované kombinácie váh je možné použiť pri sérii evolučných hľadaní (každé z nich používa iba jednu fixovanú kombináciu váh) alebo v rámci jedného evolučného hľadania vo forme dynamickej váženej agregácie parciálnych vhodností (váhy sa menia počas hľadania medzi jednotlivými generáciami).

Jednotlivé váhy je síce možné určovať náhodným výberom, avšak kvôli rovnomernosti pokrytia sa zdá byť výhodnejšou periodická zmena váh [17]. Pre agregáciu dvoch parciálnych vhodností je možné použiť

$$\begin{aligned} w_1(t) &= \mid sin(2\pi t/K) \mid \\ w_2(t) &= 1 - w_1(t) \end{aligned} \tag{2.9}$$

kde $\mid . \mid$ reprezentuje absolútnu hodnotu, t je aktuálna generácia a K konštanta určujúca rýchlosť zmien. V tomto prípade sa každá váha postupne mení tak, že periodicky prechádza od jedného extrému k druhému – od neuvažovania príslušnej parciálnej vhodnosti ($w_i = 0$) k uvažovaniu iba danej parciálnej vhodnosti ($w_i = 1$) a naopak. Čím je hodnota K menšia, tým sú zmeny rýchlejšie – zmena váhy z jedného extrému do druhého trvá $K/4$ generácií. Frekvencia zmien by nemala byť príliš veľká aby algoritmus bol schopný konvergencie k aktuálnemu extrému.

Iný prístup k nastavovaniu váh je využívať nejaký adaptačný mechanizmus. Jednoduchým zosobnením tohto je vážiť každú parciálnu vhodnosť celkovou parciálnou vhodnosťou populácie[4] (pri minimalizácii) alebo jej obrátenou hodnotou (pri maximalizácii). Cieľom je vyváženie zlepšovania jedincov podľa jednotlivých kritérií. Ak totiž dochádza k zlepšovaniu iba podľa jedného kritéria, váha príslušnej parciálnej vhodnosti sa mení tak, že viac sa začína preferovať zlepšovanie podľa ostatných kritérií.

2.1.2 Zotrieďovanie podľa vhodností

Táto skupina sa od predchádzajúcej líši v tom, že sa vyhýba práci s konkrétnymi hodnotami parciálnych vhodností. Nie sú dôležité hodnoty vhodností v absolútnom chápaní ale iba v relatívnom – v ich vzťahu k hodnotám iných jedincov populácie.

Je to vlastne dvojkrokový proces, pozostávajúci z týchto dvoch základných krokov:

1. určenie poradia jedincov,

2. určenie agregovaného poradia.

V prvom kroku sú jedince zotriedené podľa každej parciálnej vhodnosti osobitne – pre každé kritérium sa takto získa jedno zotriedenie. Každý jedinec a_i tak namiesto vektoru vhodností $[\Phi_1(a_i), \ldots, \Phi_n(a_i)]$ získa vektor

[4] Súčet hodnôt danej parciálnej vhodnosti všetkých jedincov populácie.

poradí $[r_1(a_i), \ldots, r_n(a_i)]$, kde prvý prvok $r_1(a_i)$ je jeho poradie podľa prvého kritéria, druhý podľa druhého kritéria, atď. až $r_n(a_i)$ reprezentuje jeho poradie podľa posledného kritéria.

V druhom kroku sa pre každého jedinca príslušný vektor poradí nahradí jedným poradím. Toto môže mať agregačný charakter (napr. priemerné poradie jedinca) alebo výberový charakter (napr. medián poradí).

Taktiež je možné zohľadňovať dôležitosť jednotlivých kritérií. Pri agregačnom prístupe je možné určovať vážený priemer poradí, kde váhy vyjadrujú dôležitosť kritérií. Pri výberovom prístupe je možné vytvárať výberovú vzorku tak, že každé poradie sa v nej ocitne niekoľkokrát, pričom tento počet je proporcionálny dôležitosti daného kritéria.

Výsledkom je, že jedince sú zotriedené do akéhosi syntetického poradia, zohľadňujúceho všetky kritériá. Toto je možné jednoducho využiť tými selekčnými metódami, ktoré vychádzajú z usporiadania jedincov (turnaje, orezania, premapovacie metódy založené na zotrieďovaní).

Výhodou tohto prístupu je to, že nie je potrebné normalizovať obory hodnôt jednotlivých parciálnych vhodností, aby boli navzájom porovnateľné.

2.1.3 Vhodnosť založená na dominancii

Predchádzajúce prístupy mali tú nevýhodu, že neuvažovali všetky riešenia z Paretovej množiny rovnakým spôsobom. Preto pri tejto triede metód sa pri určovaní syntetickej vhodnosti vychádza priamo z pojmu dominancie. Táto je využívaná dvojakým spôsobom:

- tvorba lokálnych Paretových množín,
- určovanie počtu jedincov.

Prvý prístup je reprezentovaný procesom pravidelného striedania dvoch krokov: zmenšovania uvažovanej populácie jedincov a postupného vytvárania lokálnych Paretových množín pre uvažovaných jedincov. Tento postup má tvar algoritmu:

1. Vhodnosť Φ sa nastaví na hodnotu 1.
2. V rámci uvažovaných jedincov sa nájde lokálna Paretova množina (jedince, ktoré nie sú dominované žiadnym z uvažovaných jedincov).
3. Každému jedincovi z lokálnej Paretovej množiny sa priradí aktuálna vhodnosť Φ ako syntetická vhodnosť jedinca.

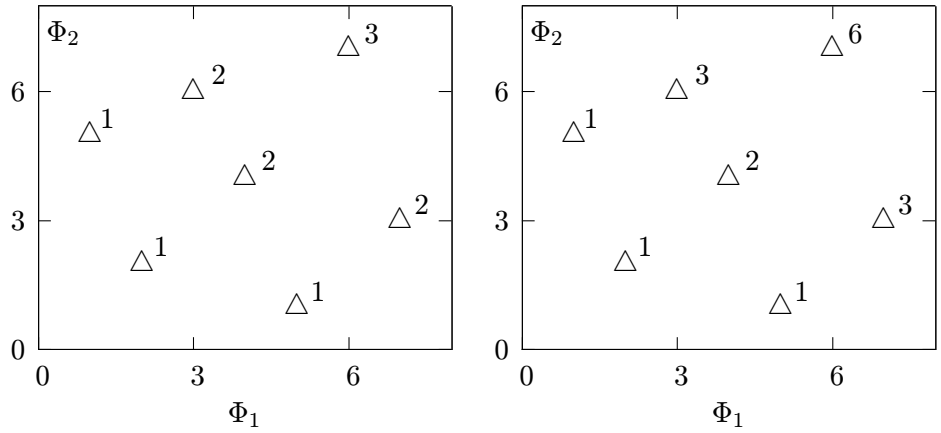
4. Z populácie sa odstránia tie jedince, ktoré už majú priradenú syntetickú vhodnosť.

5. Ak je populácia prázdna (teda všetky jedince už majú priradenú syntetickú vhodnosť), tak procedúra končí.

6. Vhodnosť Φ sa inkrementuje.

7. Návrat na bod 2.

Druhý prístup jednoducho pre každého jedinca určí, koľko jedincov z aktuálnej populácie dominuje danému jedincovi. Danému jedincovi sa potom priradí vhodnosť ktorá je o jednotku väčšia ako toto číslo. Rozdiely medzi obomi prístupmi sú ilustrované na obr. 2.3.



Obr. 2.3: Ilustrácia určenia vhodnosti na základe lokálnych Paretových množín (vľavo) a počtu dominujúcich jedincov (vpravo).

V oboch prípadoch je vhodnosť, ktorá je priraďovaná jedincom, z intervalu $< 1, \mu >$. Oba prístupy priradia rovnaké vhodnosti tým jedincom, ktoré nie sú dominované inými jedincami – teda oba prístupy poskytujú rovnakú šancu každému nedominovanému jedincovi[5]. Rozdiely sú však v zachádzaní s dominovanými jedincami. Pri použití počtu jedincov, vhodnosti majú tendenciu byť priraďované z väčšieho rozsahu ako pri lokálnych Paretových

[5]Uvedené prístupy sú vhodné vtedy, ak všetky kritériá sú považované za rovnako významné. Nie je zrejmé, ako by sa dali vhodným spôsobom využiť informácie o preferencii niektorých kritérií voči iným kritériám.

množinách – dochádza k jemnejšiemu rozlišovaniu jedincov. Následkom toho dva jedince, zdieľajúce tú istú lokálnu Paretovu množinu, môžu byť uvažované rôzne a môžu im byť pridelené rôzne hodnoty vhodnosti.

Oba prístupy v uvedenom tvare predstavujú minimalizačnú úlohu – čím je syntetická vhodnosť menšia, tým je jedinec považovaný za lepšieho. Je možné použiť aj varianty v tvare maximalizačnej úlohy. Pri prvom prístupe stačí v prvom kroku vhodnosť inicializovať nie na jednotku ale na dostatočne vysokú hodnotu a v šiestom kroku ju dekrementovať. Pri druhom prístupe je možné určovať vhodnosť nie ako inkrementovaný počet dominujúcich jedincov ale o tento počet znížiť dostatočne vysokú základňu.

Pri praktickom použití má uvedený spôsob priraďovania syntetickej vhodnosti jednu nevýhodu – jedna hodnota môže byť priradená väčšiemu množstvu jedincov (ako ilustruje obr. 2.3, náchylnejšie na toto je použitie lokálnych Paretových množín). Následkom toho je častý prípad, keď je potrebné rozhodnúť medzi jedincami s rovnakou hodnotou vhodnosti (teda medzi jedincami, kde žiaden z nich nedominuje inému) a vybrať jedného z nich.

Cieľom pri tomto rozhodovaní je, aby algoritmus neskonvergoval iba do nejakej ohraničenej oblasti Paretovej množiny ale aby boli objavené riešenia zo všetkých oblastí Paretovej množiny (rovnomerne rozdelené v priestore vhodností pozdĺž celej oblasti reprezentovanej Paretovou množinou). Preto pri výskyte viacerých navzájom nedominantných jedincov s rovnakou vhodnosťou sa uprednostňujú jedince z oblastí, ktoré sú menej zastúpené, oproti jedincom z viac zastúpených oblastí.

Tento princíp za zvykne realizovať pomocou jedného z týchto dvoch spôsobov:

- úpravou vhodnosti,
- použitím dodatočného kritéria.

Prvý spôsob reprezentuje metóda zdieľania – vhodnosť jedincov je penalizovaná v závislosti na prítomnosti iných jedincov v ich blízkosti. Čím je týchto blízkych jedincov viac resp. sú bližšie, tým je penalizácia vhodnosti daného jedinca väčšia. Jedná sa o rovnakú metódu ako bola popísaná v kapitole 1.1.1. Jediný rozdiel je v tom, že teraz sa vzdialenosť dvoch jedincov neurčuje ako vzdialenosť v priestore prehľadávania ale ako vzdialenosť v priestore vhodností podľa vzťahu

$$d(a_i, a_j) = \sqrt[k]{\sum_{l=1}^{n} (\mid \Phi_l(a_i) - \Phi_l(a_j) \mid)^k} \tag{2.10}$$

ktorý pre $k = 2$ reprezentuje Euklidovu vzdialenosť. Je však možné použiť aj $k < 2$ pre rozdielnu penalizáciu v rozličných smeroch[6].

Príkladom dodatočného kritéria je zavedenie faktoru stlačenia (angl: squeeze factor). Priestor vhodností sa rozdelí na oblasti (pri dvoch vhodnostiach na obdĺžniky) tak, že každá os sa podelí na sekvenciu intervalov zadanej dĺžky. Faktor stlačenia nejakého jedinca je potom počet tých jedincov, ktoré sa nachádzajú v rovnakej oblasti ako daný jedinec[7]. Preferuje sa menšia hodnota – ak viaceré jedince majú rovnakú vhodnosť, tak sa vyberie ten, ktorý má najmenšiu hodnotu faktoru stlačenia.

Iným príkladom je zavedenie veľkosti okolia (angl: crowding distance), ktoré má jedinec k dispozícii v priestore vhodností (veľkosť prázdneho priestoru obklopujúceho jedinca). Veľkosť okolia sa určuje pre každú parciálnu vhodnosť osobitne. Ak sa určuje pre jedinca a_i s ohľadom na vhodnosť Φ_j, tak najprv sa všetky jedince zoradia podľa vhodnosti Φ_j a potom sa veľkosť okolia určí ako

$$\mid \Phi_j(a_{i+1}) - \Phi_j(a_{i-1}) \mid \tag{2.11}$$

kde a_{i-1} reprezentuje jedinca nachádzajúceho sa vo vytvorenom poradí tesne pred jedincom a_i a a_{i+1} zase reprezentuje jedinca nachádzajúceho sa v danom poradí tesne za jedincom a_i. Takýmto spôsobom sa určí veľkosť okolia s ohľadom na každú parciálnu vhodnosť a výsledná hodnota sa určí ako priemer hodnôt pre jednotlivé parciálne vhodnosti. Preferuje sa väčšia hodnota – ak viaceré jedince majú rovnakú vhodnosť, tak sa vyberie ten, ktorý má k dispozícii najväčšie okolie.

Alternatívou k veľkosti okolia môže byť vzdialenosť jedinca k nejakému inému jedincovi – napríklad k najbližšiemu. Je možné aj zovšeobecnenie. V tomto prípade sa ostatné jedince najprv zoradia podľa vzdialenosti od daného jedinca (od najbližšieho po najvzdialenejšieho) a z takto zoradených jedincov sa vyberie j-ty v poradí (pre $j = 1$ to bude najbližší, často sa však používa j rovné druhej odmocnine počtu jedincov). A vzdialenosť vybratého j-teho jedinca k danému jedincovi sa použije ako dodatočné kritérium, pričom sa preferuje väčšia hodnota.

V predchádzajúcom pri snahe dosiahnuť dva ciele (minimalizovať vzdialenosť ku globálnej Paretovej množine a maximalizovať diverzitu nedominovaných jedincov) sa dominancia dopĺňala ďalším dodatočným princípom. Novšie prístupy založené na indikátoroch [48] túto kombináciu robiť nemu-

[6]Paretova množina v priestore vhodností reprezentuje diagonálne orientovaný útvar. Pre $k < 2$ sú jedince penalizované v smere niektorej z osí do väčšej vzdialenosti ako v diagonálnom smere.

[7]Všetky jedince z jednej oblasti majú rovnakú hodnotu faktoru stlačenia.

sia, pretože indikátory umožňujú formalizovať spojité zovšeobecnenie relácie dominancie a teda nie je nutné dopĺňať diskrétnu reláciu dominancie o dodatočnú spojitú informáciu.

2.2 Metódy používajúce vektor vhodností

Do tejto skupiny je možné zaradiť všetky tie metódy, ktoré akceptujú existenciu vektora vhodností a snažia sa s ním pracovať bez toho, aby ho transformovali na jednu syntetickú hodnotu. Inak povedané, namiesto modifikácie vektora vhodností modifikujú tie časti evolučného algoritmu, ktoré pracujú s vhodnosťou jedincov.

Je možné rozoznať dve podskupiny metód podľa toho, či sa súčasne uvažuje:

- jedna vhodnosť,
- viac vhodností.

Reprezentantom prvej skupiny je VEGA (Vector Evaluated Genetic Algorithm) [39], ktorý je pravdepodobne prvým pokusom v oblasti hľadania nedominovaných riešení.

Tento algoritmus je založený na zmene selekcie rodičov. Počet rodičov ϱ, ktorých je potrebné pomocou selekcie vybrať, sa rozdelí na toľko častí, koľko zložiek sa vyskytuje vo vektore vhodností. Pri selekcii i-tej skupiny rodičov sú jedince selektované z populácie na základe svojej i-tej parciálnej vhodnosti.

Po samotnej selekcii jedincov do úlohy rodičov sa pridáva ešte jedna dodatočná operácia – všetky selektované jedince sú náhodne premiešané. Cieľom tohto kroku je odstrániť usporiadanie rodičov do skupín podľa jednotlivých kritérií.

Ak sa v úlohe selekčnej metódy používa proporcionálna selekcia, tak potom sa očakáva, že nejaký jedinec a_i, ktorému prislúcha vektor vhodností $[\Phi_1(a_i), \Phi_2(a_i), \ldots, \Phi_n(a_i)]$, vyprodukuje očakávaný počet rodičov

$$\eta(a_i) = \sum_{j=1}^{n} \frac{\varrho}{n} \frac{\Phi_j(a_i)}{\sum_{l=1}^{\mu} \Phi_j(a_l)} \tag{2.12}$$

kde μ je veľkosť populácie. Podiel ϱ/μ znamená, že podľa každej vhodnosti sa vyberal rovnaký počet rodičov[8] – a teda nie je žiadna preferencia medzi jednotlivými kritériami. Preferenciu je možné dosiahnuť rôznym počtom rodičov vyhradeným výberom podľa rôznych kritérií.

[8] Veľkosť populácie sa z praktických dôvodov volí tak, aby daný počet bol celočíselný.

Uvedená selekcia preferuje extremálne jedince – tie jedince, ktoré sú veľmi dobré aspoň podľa jedného kritéria. Tie jedince, ktoré síce patria do Paretovej množiny ale sú kompromisné (nie sú veľmi dobré ale ani veľmi zlé podľa žiadneho kritéria), sú potláčané kvôli preferovaniu extremálnych jedincov (a navyše miera ich potláčania závisí od tvaru Paretovej množiny – iné potláčanie je pre konkávny tvar a iné pre konvexný). Teda nie je rovnaké zaobchádzanie pre všetky nedominované jedince.

Iným prístupom je priamo pracovať s vektorom vhodností. Použitie dominancie vlastne umožňuje priame porovnanie takýchto dvoch vektorov a rozhodnúť, ktorý z nich je možné považovať za lepší (ktorý dominuje druhému). Na tomto princípe môže byť založená každá selekčná metóda, ktorá používa vhodnosť jedincov iba na ich vzájomné porovnanie (napr. na výber jedného z nich alebo na zotriedenie jedincov).

Príkladom môže byť modifikácia binárneho turnaja. Náhodne sa zo skupiny jedincov vyberú dva jedince a ich vektory vhodností sa porovnajú. Ak jeden dominuje druhému, je príslušný jedinec považovaný za lepšieho a stáva sa rodičom. Ak medzi vybratými jedincami nie je vzťah dominancie, sú oba považované za rovnocenné (a rodičom sa stáva jeden z nich vybratý náhodným spôsobom).

Nevýhodou môže byť to, ak sa často stáva, že ani jeden z vybratých jedincov nie je dominantný voči druhému. Pre zníženie počtu takýchto výberov je možné okrem dvoch jedincov (kandidátov) náhodne vybrať aj porovnávaciu vzorku niekoľkých ďalších jedincov. Ak na základe vzájomnej dominancie sú obaja kandidáti rovnocenní, tak sa určuje, ktorý z nich dominuje väčšiemu počtu jedincov z porovnávacej vzorky (alebo ktorý z nich je dominovaný menším počtom jedincov z porovnávacej vzorky).

Ak ani toto rozširujúce porovnanie neumožňuje rozhodnúť, je možné použiť nejaké dodatočné kritérium, napr. faktor stlačenia, veľkosť okolia, a pod.

Náhrada býva často realizovaná ako redukcia skupiny jedincov na menší počet – výber bez náhrady, kde každý jedinec je vybratý maximálne raz. Toto môže byť založené na snahe vybrať čo najviac tých jedincov, ktoré vytvárajú Paretovu množinu.

Pri takejto forme náhrady sa pôvodná skupina jedincov rozdelí na dve časti – na jedince, ktoré nie sú dominované žiadnym iným jedincom, a jedince, ktoré sú dominované aspoň jedným iným jedincom. Ak je voľných miest (počet miest μ_{t+1} v nasledujúcej generácii $P(t+1)$) viac ako je jedincov v prvej skupine, tak sa vyberú všetky jedince z prvej skupiny a ostávajúce

voľné miesta sa zaplnia náhodným výberom z druhej skupiny[9]. Ak je počet miest menší ako je počet jedincov v prvej skupine, tak sa dominované jedince zahodia a výber bude prebiehať iba zo skupiny nedominovaných jedincov. Samotný výber bude prebiehať tak, že tie nedominované jedince, ktoré ešte neboli vybraté, sa postupne zotriedia podľa každej z n parciálnych vhodností a pri každom zotriedení sa vyberie

$$\frac{\mu_{t+1}}{n+\varepsilon} \tag{2.13}$$

najlepších jedincov. Zväčšenie menovateľa o malé číslo $\varepsilon > 0$ má za následok, že po skončení výberov podľa jednotlivých kritérií ešte stále budú k dispozícii voľné miesta. Tieto sa zaplnia náhodným výberom z dosiaľ nevybraných nedominovaných jedincov (ktoré by mali reprezentovať kompromisné riešenia, keďže nepatrili medzi najlepšie jedince podľa žiadnej parciálnej vhodnosti).

2.3 Rozšírenie štruktúry algoritmu

Paretova množina je typicky zložená z veľkého počtu rovnocenných riešení, preto je účelné získať čo najväčšiu vzorku. Počet riešení, ktoré je možné získať, je však limitovaný použitou veľkosťou populácie. Je síce možné túto veľkosť zvýšiť, to však má za následok zvýšené nároky na výpočtové zdroje.

Riešením je použitie dodatočnej pamäte nájdených riešení – externej populácie. Zatiaľ čo normálna populácia (nazývaná aj interná) slúži na realizáciu evolučného hľadania, externá populácia hrá úlohu archívu. Veľkosť tohto archívu limituje počet riešení, ktoré je možné objaviť – tento počet je potom nezávislý na veľkosti populácie. Po ukončení evolučného hľadania sa obsah externej populácie považuje za nájdené riešenia.

Počas hľadania dochádza k aktualizácii archívu. Najjednoduchšou aktualizačnou schémou je jeho aktualizácia v každej generácii. Táto schéma pozostáva z dvoch základných krokov:

- výber kandidátov,
- zaradenie kandidátov do archívu.

Kandidáti sa vyberajú spomedzi jedincov aktuálnej internej populácie. Tento výber je tak isto dvojkrokovým procesom:

[9]Samozrejme, je možné namiesto náhodného výberu určovať lokálne Paretove množiny a tie postupne prenášať.

1. v rámci internej populácie sú identifikované nedominované jedince (jedince, ktoré nie sú dominované žiadnym iným jedincom internej populácie),

2. zo skupiny identifikovaných kandidátov sa vyradia tie jedince, ktoré sú dominované aspoň jedným jedincom z externej populácie.

Vybraní kandidáti patria medzi najlepšie doteraz nájdené riešenia – počas evolučného hľadania nebol nájdený žiadny jedinec, ktorý by bol v zmysle dominancie lepší.

Títo kandidáti sú postupne zaraďovaní do archívu. Pre každého z nich sa opakuje postup:

1. všetky jedince v archíve sú skontrolované, či nie sú dominované daným kandidátom. Tie jedince, ktoré sú dominované, sú z archívu vylúčené.

2. ak v archíve je voľné miesto, tak daný kandidát je vložený do archívu na jedno z voľných miest.

Po určitej dobe evolučného hľadania môže nastať prípad, že kapacita archívu už nepostačuje a nie sú v ňom voľné miesta. Vtedy sa buď vkladanie nerealizuje alebo je nutné aktivovať procedúru vytvárajúcu voľné miesta pre novoobjavených kandidátov. V druhom prípade sa buď noví kandidáti vložia do archívu a ten sa následne redukuje na prípustnú veľkosť alebo najprv sa z archívu odstráni potrebný počet jedincov a na uvoľnené miesta sa vložia noví kandidáti.

V oboch prípadoch rozhodnutie nie je možné založiť na vhodnosti jedincov, pretože všetky jedince (kandidáti aj členovia archívu) sú rovnako dobrými riešeniami – ani jeden nie je dominovaný iným. Rozhodnutie musí spočívať na nejakej inej charakteristike (napríklad na vzdialenosti medzi jedincami, na hodnote faktora stlačenia a pod.).

2.4 Porovnanie metód pre viackriteriálne úlohy

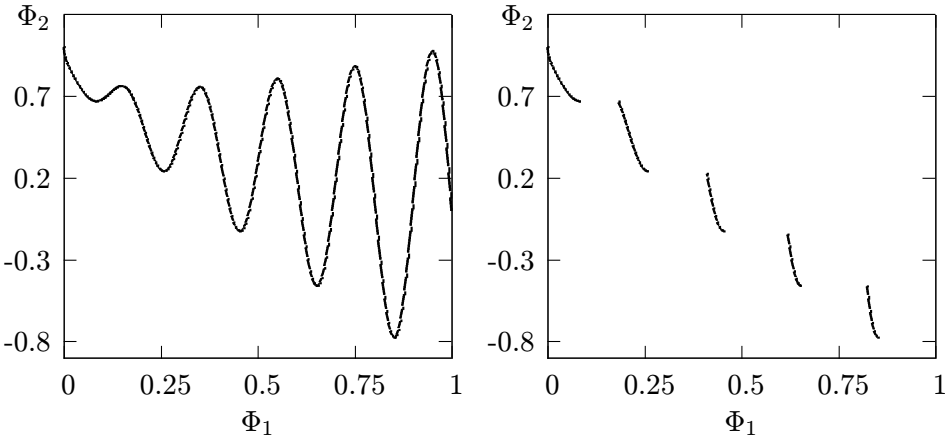
Pre empirické porovnanie vybraných metód pre multikriteriálnu optimalizáciu sme použili variáciu úlohy z [4]. Jedná sa o dvojkriteriálnu úlohu n premenných $x_1, \ldots, x_n$ s nespojitou globálnou Paretovou množinou, pričom je potrebné minimalizovať tieto parciálne vhodnosti:

$$\begin{aligned} \Phi_1(\vec{x}) &= x_1 \\ \Phi_2(\vec{x}) &= g(\vec{x}) \left(1 - \sqrt{\tfrac{x_1}{g(\vec{x})}} - \tfrac{x_1}{g(\vec{x})} sin(10\pi x_1)\right) \end{aligned} \tag{2.14}$$

kde $g(\vec{x})$ reprezentuje pomocnú funkciu tvaru

$$g(\vec{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1} \tag{2.15}$$

Pri všetkých realizovaných experimentoch každá premenná bola skúmaná v intervale $< 0, 1 >$.

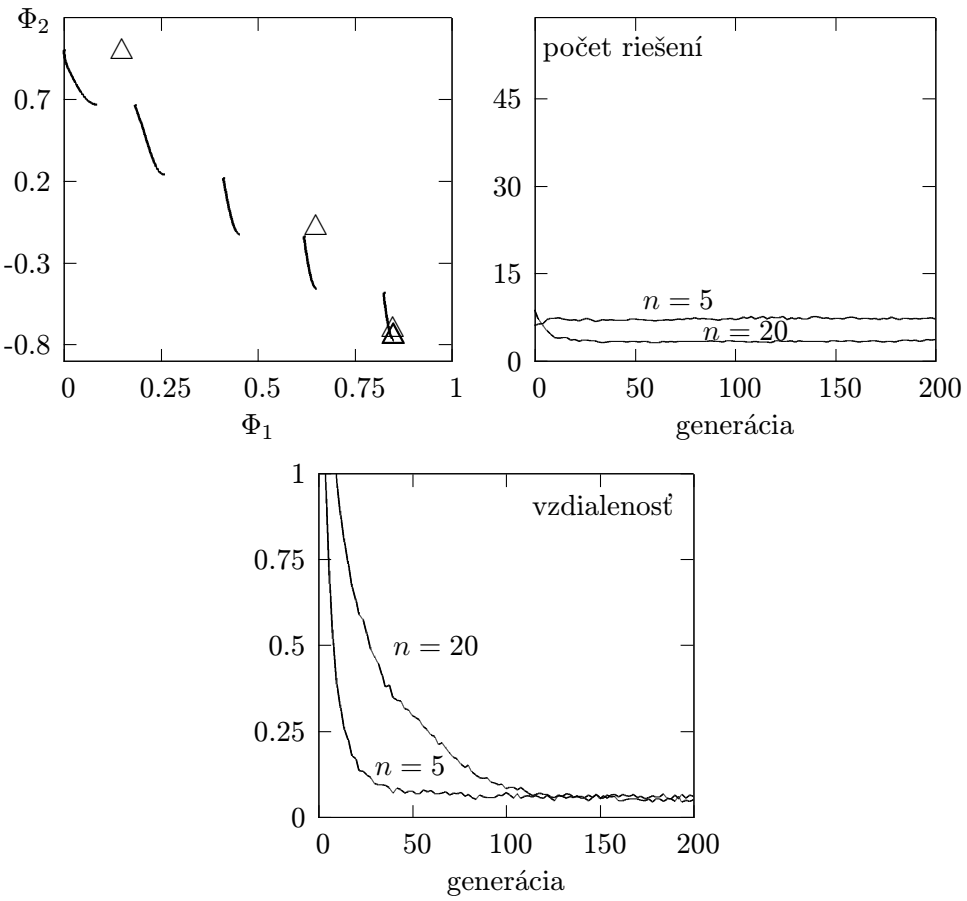


Obr. 2.4: Hranica oblastí (vľavo) a poloha Paretovej množiny (vpravo).

Funkcia vhodnosti Φ_2 je periodická vzhľadom na hodnotu premennej x_1 a tým vlastne aj vzhľadom na funkciu vhodnosti Φ_1. Toto je ilustrované na obr. 2.4, ktorý vizualizuje časť priestoru vhodností. Ľavá strana zobrazuje krivku ohraničujúcu oblasť výskytu jedincov (tie sa nachádzajú nad danou krivkou, pod ňou sa nenachádza žiadny jedinec). Vzhľadom na potrebu minimalizovať obe parciálne vhodnosti, Paretova množina má tvar podľa pravej časti obrázku. Je zrejmé, že globálne riešenia vytvárajú päť navzájom oddelených segmentov.

Ako testovacie prostredie bol použitý jednoduchý tvar testovacieho algoritmu, ktorý využíval nasledujúce prvky:

- Grayov binárny kód pre reprezentáciu hodnôt premenných (každá premenná bola reprezentovaná pomocou 12 bitov),
- náhodnú inicializáciu populácie (o veľkosti 50) jedincov,
- selekciu 50 rodičov binárnym turnajom s náhradou,



Obr. 2.5: Výsledky pre testovací algoritmus.

- tvorbu potomkov dvojbodovým krížením ($p_c = 0.5$) a mutáciou zmeny bitu ($p_m = 1/12n$),
- generačnú náhradu jedincov populácie novo vytvorenými potomkami.

Štruktúra algoritmu umožňovala tento základný tvar rozširovať o rôzne metódy pre spracovanie viacerých vhodností. K dispozícii bol aj archív pre 50 jedincov použiteľný pre tie metódy, ktoré využívajú externú populáciu.

Chovanie testovacieho algoritmu (používajúceho agregovanú vhodnosť reprezentovanú neváženou sumou oboch parciálnych vhodností) je na obr. 2.5. Sledoval sa priebeh počtu nájdených nedominovaných riešení (vpravo hore) a vzdialenosti nájdených nedominovaných riešení k najbližšiemu členovi glo-

bálnej Paretovej množiny (dole). Tieto priebehy vyjadrujú priemerné hodnoty – bolo vykonaných 200 opakovaní a dosiahnuté výsledky boli spriemernené. Pre zisťovanie chovania pre rôznu dimenzionalitu problému boli vykonané experimenty pre 5 a 20 premenných[10].

Toto spriemernené chovanie je doplnené zobrazením nájdených nedominovaných riešení v priestore riešení (na obrázku vľavo hore), ktoré ilustruje distribúciu týchto nájdených riešení voči hľadaným globálnym riešeniam[11]. Daná ilustrácia je založená na jednej realizácii hľadania riešení[12] pre kompromisný počet premenných $n = 10$.

Otázkou je, ako by sa mali uvedené rozloženie a časové priebehy zmeniť v prípade "dobrej" metódy viackriteriálnej optimalizácie. Je možné očakávať:

- zvýšenie počtu nájdených nedominovaných riešení (maximálny možný počet je 50),
- väčší pokles vzdialenosti nájdených riešení ku globálnym riešeniam (ideálne pokles na nulovú hodnotu),
- distribúcia riešení v priestore vhodností tak, aby všetkých päť zobrazených segmentov bolo vzorkovaných rovnomerným spôsobom.

Do použitého testovacieho algoritmu boli pridávané tieto metódy pre riešenie viackriteriálnych úloh:

EDWA – Evolutionary Dynamic Weighted Aggregation [17]

Používa agregáciu parciálnych vhodností do jednej skalárnej hodnoty pomocou váženej lineárnej kombinácie, pričom váhy sa dynamicky menia podľa vzťahu (2.9).

Ako ďalší prvok používa externú populáciu pre archiváciu nedominovaných riešení. Táto externá populácia je aktualizovaná nedominovanými jedincami z aktuálnej populácie, pričom dominované jedince sú z externej populácie vypúšťané. Jedinec aktuálnej populácie môže byť vložený do externej populácie iba ak nie je podobný[13] žiadnemu jedincovi v externej populácii.

[10] Čím menší počet premenných, tým je úloha jednoduchšia. Preto pri nižšom počte premenných je nájdených viac riešení a ich vzdialenosť od globálnych riešení klesá rýchlejšie.

[11] Presne podľa predpokladov – pri agregácii lineárnou kombináciou jednotlivých parciálnych vhodností populácia konverguje k jednému alebo niekoľkým málo globálnym riešeniam.

[12] Keďže nebolo použité spriemerňovanie, tak nejaká iná realizácia hľadania by poskytla iné rozloženie nájdených riešení.

[13] Podobnosť je meraná ako Euklidova vzdialenosť v priestore vhodností.

Ak je externá populácia plná (a nedochádza k vypusteniu žiadneho dominovaného jedinca z nej), nepoužíva sa žiadna procedúra pre vytváranie voľného miesta, k vloženiu jedinca do externej populácie nedôjde.

MOGA – Multiobjective Optimization Genetic Algorithm [9]

Selekčná vhodnosť sa určuje na základe dominancie medzi jedincami. Používa sa prístup založený na určovaní počtu jedincov, ktoré dominujú hodnotenému jedincovi (s cieľom minimalizovať selekčnú vhodnosť).

Pre potlačenie prílišného počtu jedincov s rovnakými hodnotami vhodnosti je selekčná vhodnosť upravená zdieľaním v závislosti na Euklidovej vzdialenosti medzi jedincami v priestore vhodností (s modifikáciou reflektujúcou to, že čím menšia vhodnosť tým lepší jedinec).

NSGA – Non-dominated Sorting Genetic Algorithm [5]

Selekčná vhodnosť sa určuje na základe dominancie, pričom sa používa modifikácia prístupu založeného na tvorbe lokálnych Paretových množín.

V populácii sa identifikujú nedominované jedince (tvoriace prvú lokálnu Paretovu množinu) a každému sa priradí rovnaká dostatočne vysoká vhodnosť. Táto vhodnosť je následne modifikovaná zdieľaním, pričom sa pri určovaní Euklidovej vzdialenosti v priestore vhodností uvažujú iba jedince, ktorým práve bola priradená vhodnosť.

Nedominované jedince sa pri ďalšom postupe už neuvažujú. Spomedzi ostávajúcich jedincov sa identifikujú nedominované jedince vytvárajúce druhú lokálnu Paretovu množinu a každému z nich sa priradí rovnaká vhodnosť, ktorá je menšia ako vhodnosť najhoršieho jedinca z predchádzajúcej lokálnej Paretovej množiny. Táto je opäť upravená zdieľaním, pričom pri zdieľaní sa uvažujú iba jedince aktuálnej lokálnej Paretovej množiny.

Postup sa opakuje dovtedy, kým každý jedinec populácie nemá priradenú selekčnú vhodnosť.

NSGA-II – Non-dominated Sorting Genetic Algorithm II [6]

Selekčná vhodnosť sa určuje na základe dominancie medzi jedincami. Používa sa prístup založený na tvorbe lokálnych Paretových množín.

Selekcia sa realizuje porovnávaním selekčných vhodností vybraných kandidátov – ak sú rôzne, tak sa vyberie jedinec s menšou hodnotou. V prípade rovnosti sa ako sekundárne rozhodovacie kritérium používa veľkosť okolia, ktoré má daný jedinec v priestore vhodností k dispozícii, s ohľadom na každú parciálnu vhodnosť podľa vzťahu (2.11).

Náhrada aktuálnej populácie sa robí pomocou plusovej stratégie. Jedince aktuálnej populácie a novo vytvorení potomkovia sú uvažovaní spolu a spomedzi nich všetkých sa vyberie príslušný počet najlepších jedincov. Pred tým sú však všetky jedince zoradené, pričom pre zoradenie sa použije rovnaké kritérium ako pre selekciu (avšak teraz je nutné zohľadniť ako jedince pôvodné tak aj jedince novo vytvorené).

SPEA – Strength Pareto Evolutionary Algorithm [47]

Používa externú populáciu pre archiváciu nedominovaných riešení. Ak je externá populácia plná, pokračuje sa vo vkladaní s následnou redukciou založenou na aglomeratívnom zhlukovaní. Každý jedinec je považovaný za zhluk a iteračne dochádza k spájaniu dvoch zhlukov (spoja sa tie dva zhluky, ktorých priemerná vzdialenosť[14] je najmenšia) až pokiaľ počet zhlukov neklesne na hodnotu veľkosti archívu. Potom sa z každého zhluku ponechá iba jeden jedinec (ten, ktorého priemerná vzdialenosť k ostatným členom daného zhluku je najmenšia) a ostatné sa zahodia.

Rodičia sa vyberajú zo zjednotenia populácie a externej populácie. Selekčná vhodnosť sa určí dvojstupňovým procesom. Najprv sa určí pre členov archívu ako podiel počtu tých jedincov populácie, ktoré sú dominované daným jedincom, a veľkosti populácie zväčšenej o jednotku. Následne sa určí selekčná vhodnosť aj pre členov populácie ako o jednotku zväčšený súčet selekčných vhodností tých jedincov z archívu, ktoré dominujú danému jedincovi.

SPEA2 [49]

Používa externú populáciu avšak nielen pre nedominované riešenia. Tá sa aktualizuje tak, že obsah externej populácie sa spojí s obsahom populácie, vyberú sa nedominované riešenia a vložia sa do archívu. Ak ich je viac ako je povolená veľkosť archívu, tak sa realizuje iteratívna redukcia – v každom kroku sa vypustí ten jedinec, ktorého vzdialenosť k najbližšiemu susedovi je najmenšia (v prípade rovností sa uvažuje druhá najmenšia, atď.). Ak v archíve naopak ostali voľné miesta, tak sa doplnia najlepšími dominovanými jedincami (dominovanými jedincami s najvyššou vhodnosťou).

Selekčná vhodnosť sa určuje pre jedincov archívu aj populácie spolu bez ohľadu na to, odkiaľ jedinec je. Najprv sa pre každého jedinca určí sila ako počet tých jedincov, ktoré sú dominované daným jedincom. Potom sa

[14] Počítaná ako priemer vzdialeností všetkých dvojíc jedincov, kde jeden z dvojice je vyberaný z prvého zhluku a druhý z druhého.

pre každého jedinca určia vzdialenosti[15] k ostatným jedincom, zoradia sa a vyberie sa k-ta vzdialenosť, kde k je odmocninou počtu jedincov, a zväčší sa o hodnotu 2. Selekčná vhodnosť každého jedinca pozostáva zo súčtu dvoch zložiek. Jedna zložka je daná ako súčet síl tých jedincov, ktoré dominujú vyhodnocovanému jedincovi. Druhá zložka je daná ako prevrátená hodnota zväčšenej vybranej vzdialenosti.

Určená selekčná vhodnosť sa používa pre výber rodičov (vyberá sa iba z externej populácie) a pre doplnenie archívu v prípade voľných miest. Po skončení hľadania sú nedominované jedince z externej populácie považované za nájdené riešenia.

PESA – Pareto Envelope-based Selection Algorithm [3].

Používa externú populáciu pre archiváciu nedominovaných riešení. Táto je inicializovaná nedominovanými riešeniami, nachádzajúcimi sa v iniciálnej populácii. Počas evolučného hľadania je externá populácia aktualizovaná nedominovanými jedincami z aktuálnej populácie, pričom dominované jedince sú z externej populácie vypúšťané.

Pre rozlišovanie jedincov v externej populácii sa používa faktor stlačenia. Rodičia sú selektovaní z externej populácie, pričom faktor stlačenia slúži ako selekčná vhodnosť jedincov. Ak je potrebné vytvoriť miesto v externej populácii, vypustí sa jedinec s najvyššou hodnotou tohto faktora.

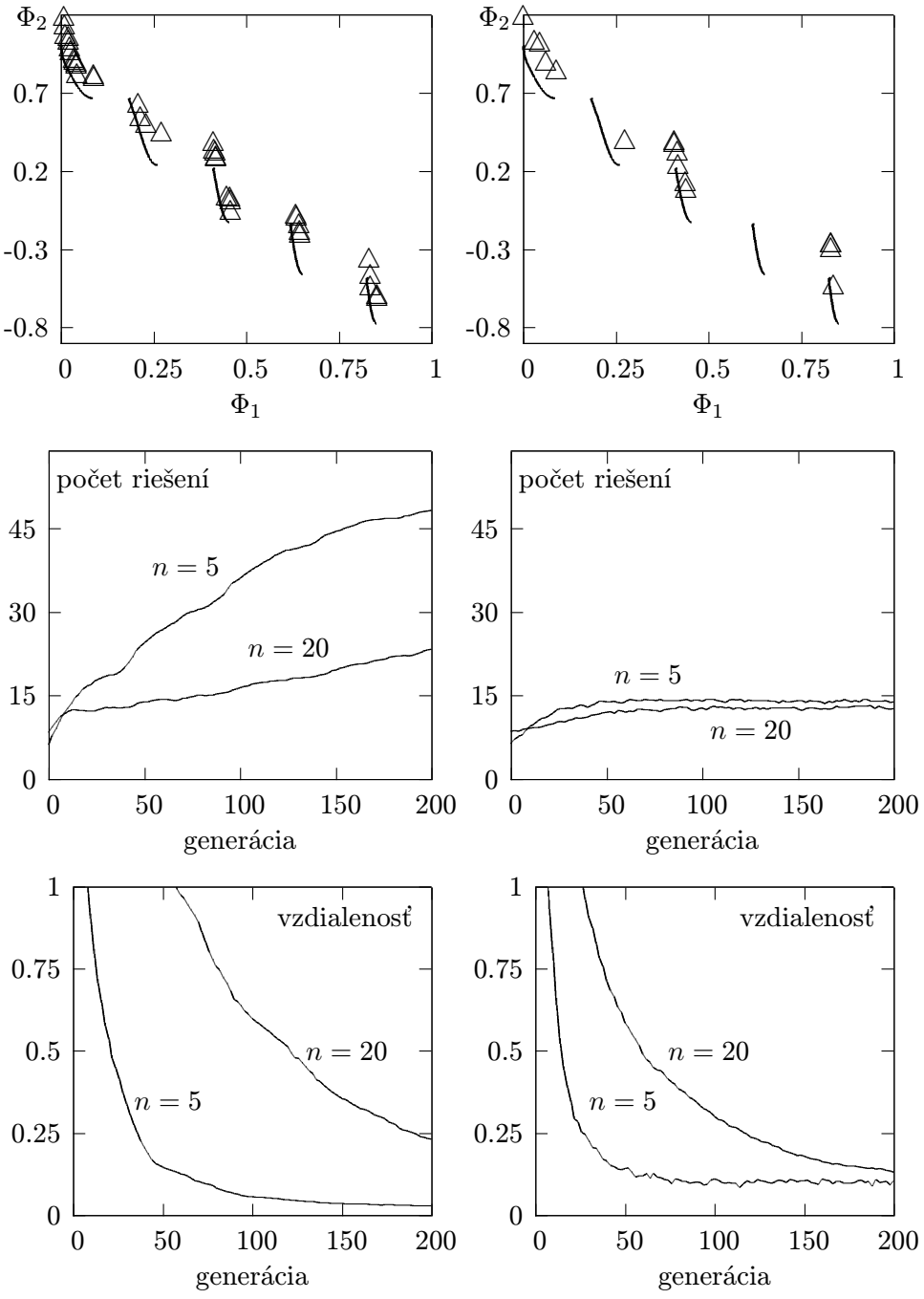
NPGA – Niched Pareto Genetic Algorithm [15]

Pracuje priamo s vektorom vhodností, pričom selekcia je realizovaná ako dominančný turnaj. Oproti pôvodnej metóde bola použitá rozšírená forma turnaja.

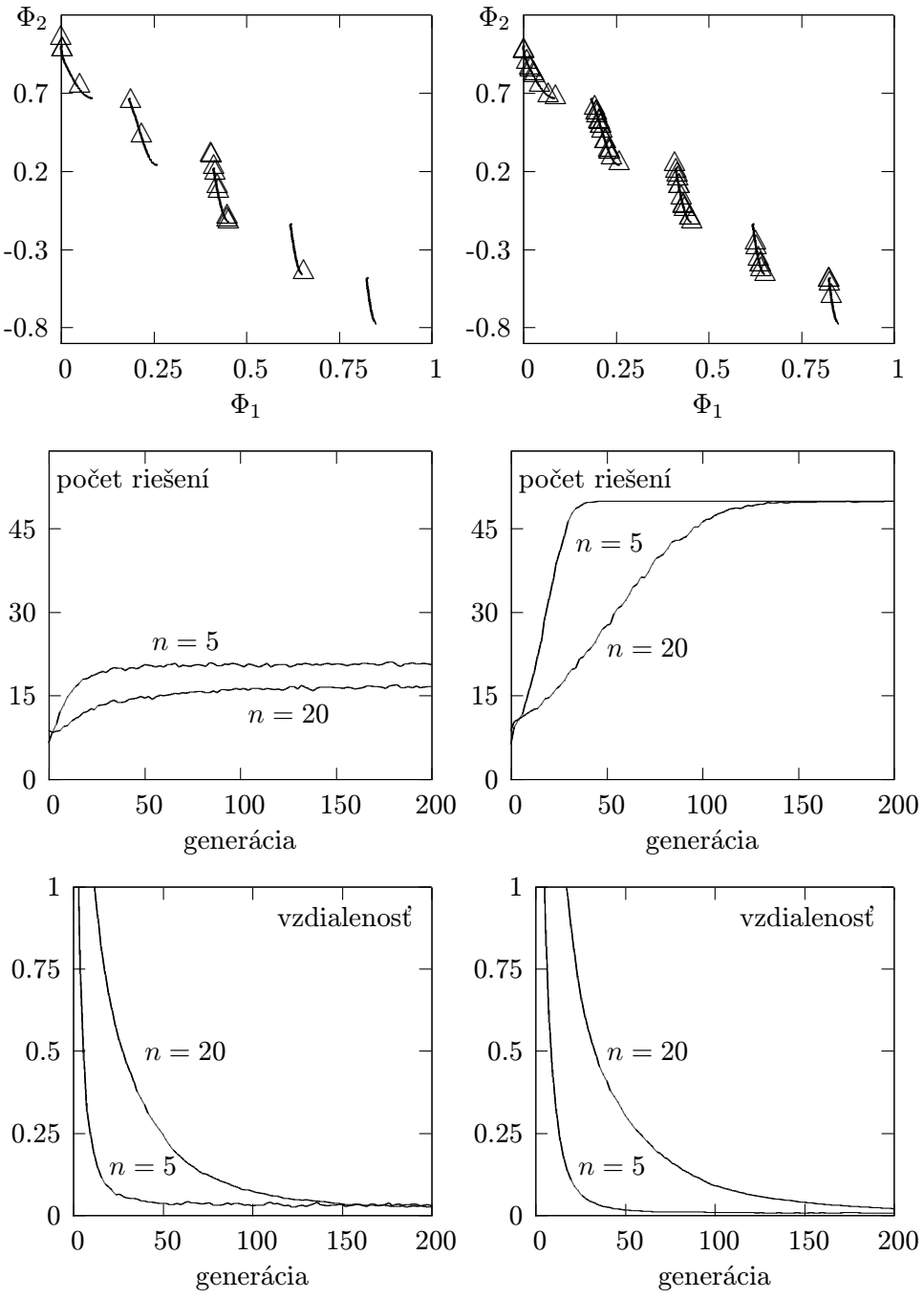
Pre selekciu každého rodiča sú z populácie náhodne vybrané dva jedince. Ich vhodnosti sú vzájomne porovnané. V prípade, že jeden vybratý jedinec dominuje druhému vybratému jedincovi, tak je považovaný za víťaza turnaja a stáva sa rodičom. Ak medzi jedincami nie je vzťah dominancie, tak je vyberaná náhodná vzorka populácie a každý z oboch kandidátov je porovnávaný voči danej vzorke. Ak jeden z kandidátov nie je dominovaný žiadnym jedincom zo vzorky a druhý je dominovaný aspoň jedným jedincom vzorky, tak nedominovaný kandidát sa stáva víťazom a rodičom. V opačnom prípade o rodičovi rozhodne náhodný výber z dvojice kandidátov[16].

[15] Euklidova vzdialenosť v priestore vhodností.

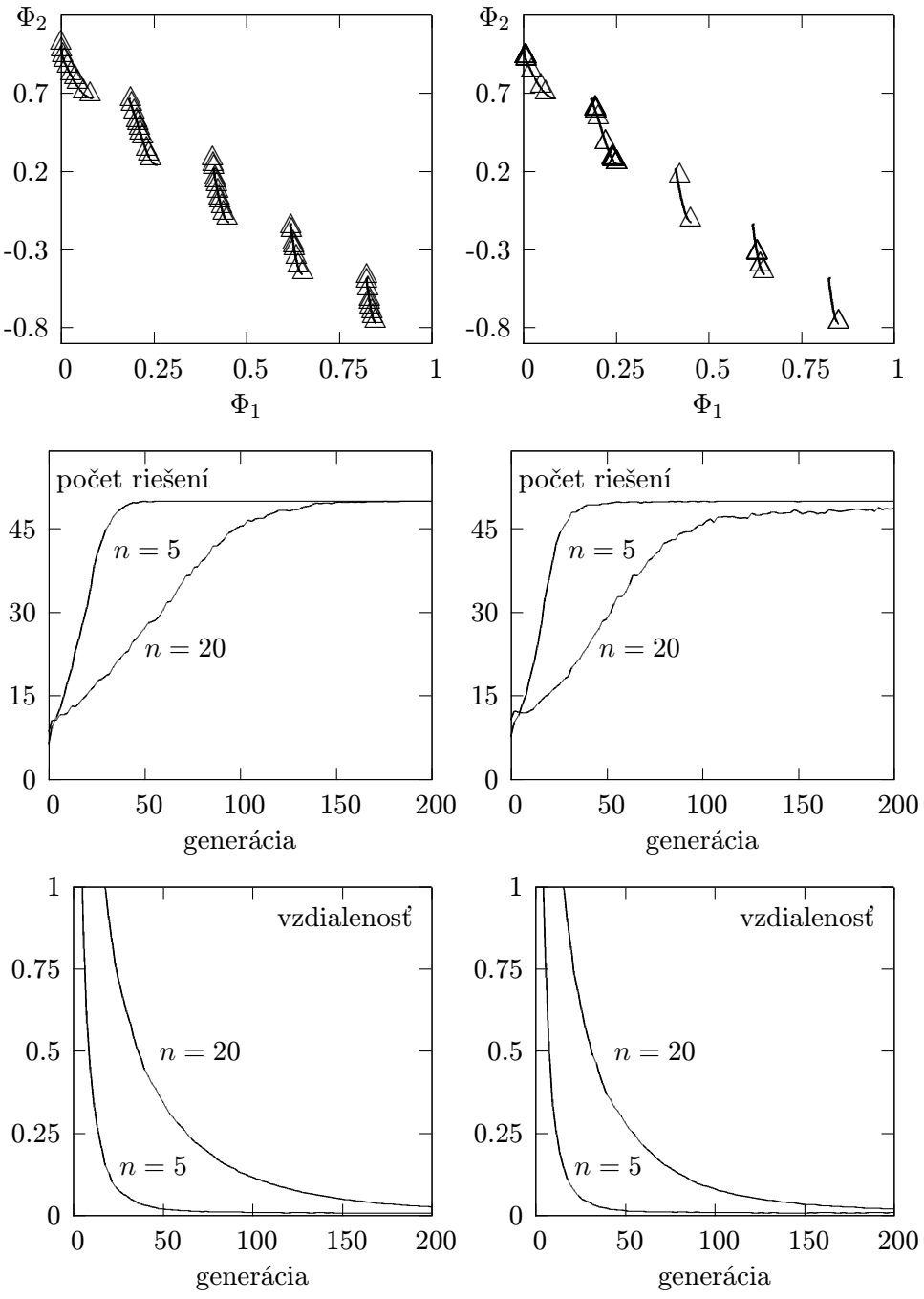
[16] Pôvodná metóda v tomto prípade používala počty jedincov, vyskytujúcich sa v okolí oboch kandidátov, pričom sa preferovala menšia hodnota.



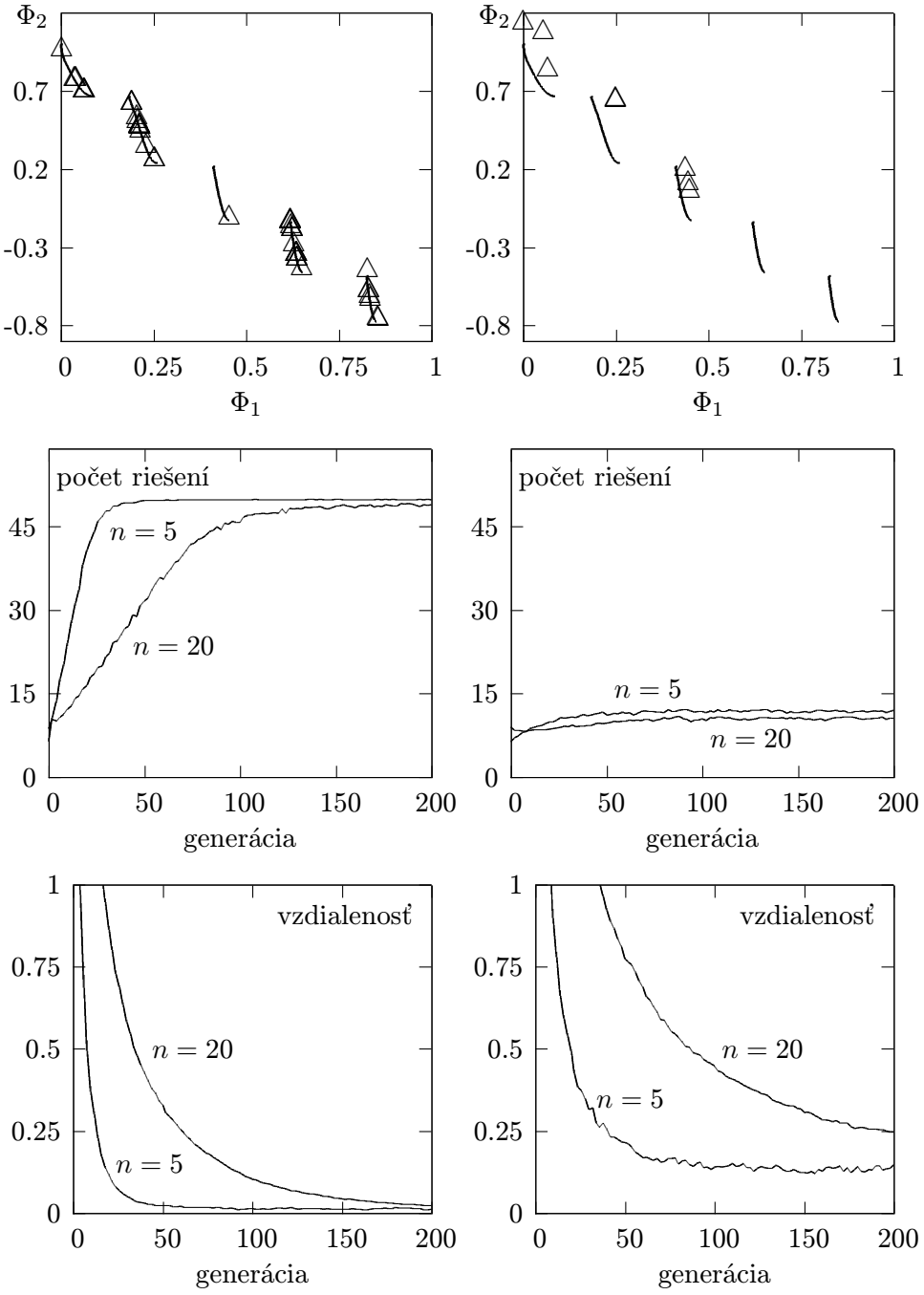
Obr. 2.6: Výsledky pre metódu EDWA (vľavo) a MOGA (vpravo).



Obr. 2.7: Výsledky pre metódu NSGA (vľavo) a NSGA-II (vpravo).



Obr. 2.8: Výsledky pre metódu SPEA (vľavo) a SPEA2 (vpravo).



Obr. 2.9: Výsledky pre metódu PESA (vľavo) a NPGA (vpravo).

Výsledky dosiahnuté uvedenými testovanými metódami sú na obr. 2.6 až 2.9. Niektoré z použitých metód nevyžadujú nastavenie žiadnych parametrov (napr. NSGA-II alebo SPEA), iné vyžadujú nastavenie viacerých parametrov (napr. NSGA alebo EDWA). V tých prípadoch, keď je toto nastavovanie potrebné, boli požadované parametre manuálne nastavané tak, aby dané metódy pre použitú testovaciu úlohu vykazovali čo najlepšie chovanie.

Metódy, ktoré využívajú externú populáciu, nemajú problém s nájdením dostatočného počtu nedominovaných riešení – PESA, SPEA a SPEA2 po pomerne krátkom čase dokážu zaplniť externý archív nedominovanými riešeniami, EDWA na to síce potrebuje viac času ale ak ho má k dispozícii, zaplní celý archív tiež. Z metód, ktoré nevyužívajú externý archív, iba NSGA-II dokáže súťažiť s metódami s externým archívom, a nájsť dostatočne veľký počet nedominovaných riešení. Ostatné metódy bez externého archívu dokážu nedominovanými jedincami zaplniť maximálne tretinu populácie, zvyšné miesta v populácii sú okupované dominovanými jedincami (a teda pri požadovanom väčšom počte nedominovaných riešení je potrebné adekvátne zväčšiť veľkosť populácie).

Tie metódy, ktoré dokážu produkovať veľký počet nedominovaných riešení, produkujú jedince nie príliš vzdialené od globálnej Paretovej množiny (iba EDWA opäť potrebuje dlhšiu dobu na dostatočné zmenšenie vzdialenosti, pretože pohyb jedincov smerom k Paretovej množine je pomalší). Metódy vytvárajúce menší počet nedominovaných riešení (MOGA, NSGA a NPGA) dosahujú väčších vzdialeností od globálnej Paretovej množiny. Dôvodom je to, že veľké množstvo dominovaných jedincov nevytvára dostatočný tlak na pohyb smerom k Paretovej množine.

Aj keď dve metódy dosahujú rovnaký priemerný počet nedominovaných riešení, môžu sa líšiť v počte riešení zobrazených v priestore vhodností (príkladom je dvojica SPEA a SPEA2). Menší počet riešení v priestore vhodností pri väčšom priemernom počte signalizuje viacnásobnú produkciu rovnakých alebo veľmi podobných riešení.

Požadovanou vlastnosťou je nielen dostatočne veľký počet rôznych nedominovaných riešení čo najbližšie globálnej Paretovej množine, ale aj rovnomernosť vzorkovania tejto Paretovej množiny. Nie je vhodné, ak niektorá jej časť je husto vzorkovaná a iná iba skromne alebo vôbec (napríklad PESA).

Z experimentov sa zdá, že pre použitú testovaciu funkciu je najlepšou voľbou NSGA-II ako reprezentant prístupu iba s internou populáciou a SPEA ako reprezentant prístupu s externou populáciou.

Kapitola 3

Riešenie nestacionárnych úloh

Pri riešení praktických úloh je nutné sa vysporiadať s rozličnými zdrojmi neurčitosti, sťažujúcimi efektívne použitie evolučných algoritmov. Jedným z nich je časová závislosť funkcie vhodnosti. Funkcia vhodnosti ostáva naďalej deterministická v každom časovom okamihu, avšak jej tvar je časovo premenlivý. Následkom týchto zmien môže byť to, že hľadaný globálny extrém (ako aj extrémy lokálne) menia svoju veľkosť a/alebo polohu. Zmena veľkosti extrému predstavuje problém iba za predpokladu, že súčasne dochádza k zmene charakteru extrému – globálny extrém sa stáva lokálnym a jeho úlohu preberá iný extrém. V tomto prípade vlastne dochádza nielen k zmene veľkosti ale aj polohy, pretože nový globálny extrém je umiestnený na inej lokácii v priestore prehľadávania, než bola poloha pôvodného globálneho extrému.

Zmena polohy znamená, že predtým nájdené riešenie prestáva byť hľadaným riešením a preto je potrebné hľadať riešenie nové. Zmeny polohy môžu byť rôzneho charakteru, avšak najčastejšie sa klasifikujú do troch rôznych tried:

- spojité zmeny – rozsahom malé avšak permanentné alebo často sa opakujúce zmeny, často sa jedná o kontinuálny posun určitým smerom,
- skokové zmeny – zmeny veľkého rozsahu, ktoré sa môžu vyskytovať v nepravidelných (relatívne dlhších) intervaloch, pričom veľkosť a smer zmeny majú náhodný charakter,
- oscilačné zmeny – skokové zmeny spôsobujúce periodické opakovanie určitého počtu polôh.

Kvôli výskytu zmien polôh preto evolučnému algoritmu nestačí jednorazovo nájsť polohu hľadaného extrému, ale mal by byť schopný sledovať jeho polohu aj naďalej a po jej zmene nájsť novú polohu daného extrému. Samozrejme, zmenený problém je možné považovať za nový problém, ktorý je nezávislý na predchádzajúcom probléme. Za takéhoto predpokladu je ho možné riešiť opätovným štartom algoritmu so zabudnutím všetkej informácie, nahromadenej v minulosti. No prijateľnejším sa zdá riešenie, ktoré dokáže zužitkovať informáciu získanú počas predchádzajúceho hľadania (alebo aspoň nejakú jej časť) pre urýchlenie opätovného nájdenia zmenenej polohy hľadaného extrému. To, či takýto prístup môže byť úspešný, závisí na charaktere zmien. Ak zmena je radikálna a zmenený problém je iba málo podobný pôvodnému problému, použitie v minulosti získanej informácie môže byť kontraproduktívne a opakovaný štart môže byť najvýhodnejším riešením.

Pokiaľ populácia jedincov obsahuje informáciu o celom priestore prehľadávania[1], tak algoritmus je dostatočne flexibilný a má šancu zareagovať na zmenu polohy hľadaného riešenia. Prirodzenou tendenciou algoritmu je však konvergencia – jedince sa sústredia iba do "najsľubnejšej" oblasti priestoru. Algoritmus po skonvergovaní do úzkeho podpriestoru okolo extrému má šancu sledovať pohyb tohto extrému iba vtedy, ak tento pohyb je pomalý a s malými zmenami (teda iba zmeny spojitého typu). Ak sa extrém náhle presunie do oblasti, v ktorej sa už žiadne jedince nenachádzajú, tak zvyčajne už algoritmus novú polohu extrému nedokáže nájsť.

Existujúce prístupy je možné klasifikovať rôznymi spôsobmi. V [16] je použité delenie do štyroch základných skupín:

- generovanie rôznorodosti po zmene – algoritmus pracuje štandardným spôsobom až do okamihu detekcie zmeny, potom je jednorazovo zvýšená rôznorodosť populácie a algoritmus pokračuje štandardne ďalej,
- udržiavanie rôznorodosti – neustále je podporované udržiavanie rôznorodosti na dostatočne vysokej úrovni,
- pamäťové prístupy – pamäť umožňuje pamätanie informácie, ktorá môže byť užitočná v prípade zmeny,
- multipopulačné prístupy – populácia je rozdelená na subpopulácie, smerujúce do rôznych oblastí priestoru prehľadávania.

Jednotiacim prvkom všetkých prístupov je zameranie sa na rôznorodosť, či už priamym alebo nepriamym (podporným) spôsobom.

[1] Príkladom je udržiavanie jedincov z rôznych častí priestoru prehľadávania v populácii.

3.1 Generovanie a udržiavanie rôznorodosti

Rôznorodosť v populácii reprezentuje prítomnosť takých stavebných blokov v populácii, ktoré algoritmu umožňujú konštruovať jedince z rôznych častí priestoru prehľadávania – čím vyššia diverzita (uniformnejšia distribúcia stavebných blokov v populácii), tým má algoritmus možnosť lepšieho pokrytia tohto priestoru. Cieľom tejto skupiny prístupov je teda pomocou vytvárania alebo udržiavania rôznorodosti populácie udržiavať algoritmus dostatočne flexibilný na to, aby bol schopný prenášať svoju pozornosť z jedného podpriestoru na iný podpriestor (v závislosti na tom, ako sa mení poloha hľadaného extrému) bez toho, aby uviazol v nejakom úzko ohraničenom podpriestore.

Relevantné prístupy sú zvyčajne založené na využívaní mutačného alebo selekčného operátora. Rozdiel je v tom, že mutačné prístupy sa snažia vkladať stavebný materiál, ktorý v populácii absentuje, naspäť do populácie, zatiaľ čo selekčné prístupy sa snažia čo najlepšie využiť ten stavebný materiál, ktorý majú k dispozícii.

3.1.1 Podporovanie rôznorodosti pomocou mutácie

Pri využívaní mutačného operátora by bolo najpriamočiarejšie jednoducho zvýšiť pravdepodobnosť mutácie. Takéto uniformné zvýšenie v celej populácii by síce malo pozitívny vplyv na rôznorodosť materiálu v populácii, avšak súčasne by sa negatívne mohlo odraziť na schopnosti algoritmu konvergovať k riešeniu v dobe, keď nedochádza k časovým zmenám funkcie vhodnosti.

Preto namiesto uniformnej distribúcie zvýšenej miery mutácie sa používa distribúcia neuniformná, a to

- neuniformná v priestore,
- neuniformná v čase.

Prvý prípad znamená, že nie každá časť populácie je mutovaná s rovnakou pravdepodobnosťou. Príkladom takéhoto prístupu je metóda *náhodných imigrantov* [13]. V tomto prípade je populácia rozdelená do dvoch častí. V rámci jednej časti sa použije rovnaká miera mutácie, aká by sa použila pre riešenie danej úlohy v jej statickej podobe. Zvýšená mutácia sa koncentruje na druhú časť populácie – táto časť je nahradená náhodne generovanými jedincami (čo je ekvivalentné pravdepodobnosti mutácie $p_m = 0.5$ pre binárnu reprezentáciu). K náhrade dochádza v každej generácii evolučného

cyklu, čo zabezpečuje stabilný prísun stavebného materiálu, ktorý mohol byť z populácie vytlačený vďaka konvergencii.

V období, keď nedochádza k zmene funkcie vhodnosti, hlavnú úlohu hrá prvá časť. V okamihu zmeny funkcie vhodnosti sa zvyšuje význam druhej časti, ktorej cieľom je umožnenie prenosu pozornosti algoritmu. Metóda má iba jeden riadiaci parameter – v akom pomere je populácia delená na obe časti[2].

Prípad neuniformnosti v čase znamená, že zvýšená mutácia nie je používaná v každej generácii, ale iba v niektorých generáciách. Typickým reprezentantom tohto prístupu je metóda *spúšťanej hypermutácie* [2]. Teraz samotný algoritmus pracuje v dvoch rôznych režimoch. V základnom režime sa používa rovnaká miera mutácie ako pri riešení statických úloh. V druhom režime (tzv. hypermutačnom) sa používa oveľa vyššia hodnota pravdepodobnosti mutácie. V oboch režimoch sa daná miera mutácie používa uniformne v rámci celej populácie.

Hypermutačný režim je inicializovaný signálom, indikujúcim zmenu funkcie vhodnosti (a teda možný presun hľadaného riešenia na inú pozíciu v priestore prehľadávania). Pre identifikáciu tejto zmeny sa monitoroval kĺzavý priemer vhodností najlepších členov populácie počas piatich posledných generácií. Ak došlo k výraznej degradácii hodnoty tohto parametra, tak bol generovaný inicializačný signál pre prechod do hypermutačného režimu.

Riadiacim parametrom metódy je pravdepodobnosť mutácie v hypermutačnom režime. Musí byť dostatočne vysoká pre zabezpečenie potrebnej rôznorodosti, avšak súčasne dostatočne nízka, aby hypermutačný režim nebol jednoduchým náhodným reštartom[3].

Príťažlivou vlastnosťou tohto prístupu je jeho adaptivita – prístup vlastne emuluje sériu použití algoritmu (v jeho štandardnej podobe) v časovo ohraničených statických prostrediach a zvýšenou pravdepodobnosťou mutácie iba reaguje na zmeny funkcie vhodnosti. Jeho nevýhodou je však to, že v niektorých prípadoch zlyháva jeho schopnosť detekcie zmeny funkcie vhodnosti – napríklad v prípade, keď nový globálny extrém vznikne v časti priestoru prehľadávania, ktorá nie je zastúpená v populácii, bez toho aby sa menila tá časť podpriestoru, do ktorej algoritmus skonvergoval.

[2] Autor metódy vo svojej práci doporučil nahrádzať 30% populácie, avšak samozrejme je možné aj použitie inej hodnoty alebo zavedenie adaptívnej schémy.

[3] Autorka hovorí o hodnotách od 0.15 pre pomalé zmeny až po 0.3 pre rýchle oscilácie.

3.1.2 Podporovanie rôznorodosti výberom

V rámci tejto skupiny je možné použiť klasické selekčné schémy, snažiace sa podporovať vyššiu úroveň rôznorodosti v populácii, ako napríklad zdieľanie alebo rôzne crowdingové schémy (napr. "najpodobnejší z najhorších" alebo "najhorší z najpodobnejších").

Tieto prístupy však pracujú s rôznorodosťou populácie iba implicitným spôsobom – aplikujú síce nejaký postup, ktorý určitou mierou podporuje udržiavanie rôznorodosti, avšak bez toho aby túto rôznorodosť explicitne kvantifikovali. Príkladom metódy explicitného zohľadňovania rôznorodosti je *termodynamický genetický algoritmus* [32]. Jeho najzaujímavejšou časťou je zostavovanie novej generácie o veľkosti μ jedincov, pričom tieto jedince sú vyberané spomedzi $2\mu+1$ jedincov (μ nových jedincov vytvorených pomocou krížiaceho operátora, μ nových jedincov vytvorených použitím mutačného operátora a jeden elitný člen pôvodnej populácie).

Pri výbere je síce použitá deterministická stratégia, avšak na rozdiel napríklad od "čiarkovej" alebo "plusovej" stratégie [26] sa nezameriava výlučne na najvhodnejšie jedince. Namiesto toho realizuje výber tak, aby minimalizoval vzťah

$$\langle -\Phi \rangle - HT \tag{3.1}$$

kde $\langle -\Phi \rangle$ značí strednú hodnotu záporne uvažovanej vhodnosti[4] a H je entropia, reprezentujúca rôznorodosť v populácii. Preferencia rôznych jedincov pri výbere je riadená parametrom T označovaným ako teplota.

Uvedený vzťah je možné minimalizovať dvomi spôsobmi – alebo zmenšovaním prvého člena (to je možné výberom jedincov s čo najväčšou vhodnosťou) alebo zväčšovaním druhého člena (to zase výberom takých jedincov, ktoré čo najviac zvyšujú entropiu – teda dodávajú do populácie stavebný materiál, ktorý v nej chýba alebo je v nej zastúpený iba v malom množstve). Vhodný pomer medzi preferovaním vhodnejších jedincov a preferovaním jedincov zvyšujúcich rôznorodosť novej populácie je nastavovaný teplotou. Zvyšovanie teploty kladie viac dôrazu na rôznorodosť, jej znižovanie zase na vhodnosť jedincov.

Samotný výber jedincov prebieha iteračným spôsobom. Na začiatku bude populácia $P(t+1) = \emptyset$ a v každom iteračnom cykle sa do nej pridá jeden jedinec – ten, ktorý aktuálne minimalizuje uvedený vzťah (3.1).

Ak predpokladáme, že počas $i-1$ iterácií už bolo do $P(t+1)$ pridaných $i-1$ jedincov, tak potom pre každého jedinca j, ktorý pripadá do úvahy pre

[4] Ak by stredná hodnota vhodnosti bola nahradená strednou energiou systému, tak uvedený vzťah by reprezentoval tzv. voľnú energiu systému.

zaradenie do $P(t+1)$ ako jej i-ty člen, sa určí hodnota

$$\langle -\Phi \rangle - TH = \frac{-\Phi(j) + \sum_{l=1}^{i-1}(-\Phi(l))}{i} - T\sum_{l=1}^{k} H_l \tag{3.2}$$

kde $\Phi(l)$ je vhodnosť l-tého člena $P(t+1)$, $\Phi(j)$ je vhodnosť skúmaného jedinca a H_l je entropia l-tej pozície použitej reprezentácie pri uvažovaní jedinca j spolu s jedincami z $P(t+1)$ – použitá reprezentácia teda používa k pozícií. Ak sa používa binárna reprezentácia, tak entropia sa určí pomocou

$$H_l = -\frac{N0}{N0+N1} log \frac{N0}{N0+N1} - \frac{N1}{N0+N1} log \frac{N1}{N0+N1} \tag{3.3}$$

kde $N0$ je počet núl na pozícii l v skúmanej skupine jedincov a $N1$ zase počet jednotiek na pozícii l v danej skupine.

Vo vzťahu (3.2) teda prvý člen určuje, aká by bola stredná vhodnosť $P(t+1)$, ak by sme do nej pridali jedinca j, a druhý člen zase určuje, aká by bola entropia $P(t+1)$, ak by sme do nej pridali jedinca j.

Teplota T môže byť konštantou, nastavenou na začiatku algoritmu. Keďže však pri takomto nastavení algoritmus nemusí byť schopný vhodnej reakcie na zmeny funkcie vhodnosti (napríklad prehnaný dôraz na entropiu môže brániť algoritmu v konvergencii do oblasti výskytu riešenia), s výhodou je možné použiť nejakú adaptívnu schému pre prispôsobovanie hodnoty teploty aktuálnemu stavu hľadania. Príkladom môže byť [33], kde teplota T sa určí pre generáciu t podľa

$$log\ T(t) = log\ T(t-1) + K(H^* - H) \tag{3.4}$$

kde H je entropia aktuálnej populácie, H^* je požadovaná entropia a K je konštanta. Ak $H > H^*$ (teda aktuálna entropia je dostatočne veľká), tak podľa uvedeného vzťahu sa teplota zníži (a viac sa bude zohľadňovať vhodnosť jedincov). Ak bude naopak $H < H^*$ (aktuálna entropia nedosahuje požadovanú úroveň), tak teplota sa zvýši (a viac sa bude zohľadňovať príspevok jedincov k rôznorodosti).

3.2 Použitie pamäti

Základná myšlienka tejto triedy prístupov je doplniť štandardný evolučný algoritmus nejakou formou pamäti, ktorá by mu umožňovala prístup k užitočnej informácii, získanej z minulých generácií. Doplnená pamäť môže byť dvojakého charakteru:

- explicitná pamäť,
- implicitná pamäť.

3.2.1 Algoritmy s explicitnou pamäťou

Explicitná pamäť môže doplniť štandardnú podobu evolučného algoritmu o pamäťový priestor pre ukladanie informácie v nejakej situácii a výber tejto informácie v nejakej inej situácii. Pamäťový priestor je riadený určitou explicitnou stratégiou, umožňujúcou jeho vhodné využitie. Takýchto stratégií principiálne existuje viacero, avšak každá z nich musí riešiť tri základné otázky:

- aká informácia sa ukladá v pamäti,
- ako sa pamäť inicializuje a aktualizuje,
- ako sa v pamäti uložená informácia využíva.

V najjednoduchšom prípade sa do pamäti ukladajú priamo konkrétne jedince – vzorky priestoru prehľadávania. Takýmto spôsobom algoritmus aj v situácii, keď jeho populácia skonverguje do úzko ohraničeného podpriestoru, má k dispozícii informáciu aj z iných častí priestoru prehľadávania, ktorá mu umožní v prípade potreby obohatiť aktuálnu populáciu a tým zvýšiť jej rôznorodosť. Prístup založený na priamom pamätaní jedincov sa označuje ako *priama pamäť* [46].

Pri jednoduchej podobe takejto pamäti je táto pamäť inicializovaná na začiatku hľadania súčasne s inicializáciou populácie. Buď je zaplnená novými náhodne generovanými jedincami alebo jedincami, ktorými bola inicializovaná populácia. Aktualizačný mechanizmus je postavený na princípe, že vždy je v pamäti vytipovaný jeden jedinec, ktorý je nahradený nejakým jedincom, vybraným z aktuálnej populácie.

Pri tejto aktualizácii pamäti snahou môže byť v pamäti udržiavať jedince s nadpriemernou vhodnosťou, jedince nie príliš staré alebo jedince distribuované v rámci viacerých podpriestorov priestoru prehľadávania. Na dosiahnutie tohto je možné z aktuálnej populácie vyberať najlepšieho jedinca a použiť jednu z aktualizačných stratégií, kde z populácie vybraný jedinec nahradí v pamäti:

- najmenej dôležitého jedinca, kde dôležitosť sa určuje ako lineárna kombinácia[5] veku jedinca, jeho vhodnosti a jeho príspevku k variancii ob-

[5]Pri kombinovaní je možné použiť aj nulovú váhu, a teda zohľadniť iba niektoré z podmienok.

sahu pamäti,

- jedinca, ktorý je najpodobnejší jedincovi vkladanému do pamäti (môže byť podmienené tým, aby vkladaný jedinec mal lepšiu vhodnosť ako jedinec v pamäti),
- menej vhodnejšieho jedinca z takého páru jedincov, v ktorom je vzdialenosť medzi členmi páru najmenšia.

V prípade, že je identifikovaná zmena funkcie vhodnosti (pokles vhodnosti jedincov v populácii), tak najlepšie jedince z pamäti (podľa ich aktuálnej vhodnosti, nie tej s ktorou boli vkladané do pamäti) jednoducho nahradia najhoršie jedince v populácii. Alternatívou je to, že nová populácia sa vytvorí z najlepších jedincov zjednotenia aktuálnej populácie a obsahu pamäti. Pri použití jedincov uchovávaných v pamäti nedochádza k zmene obsahu samotnej pamäti, iba k zmene populácie.

Pri zložitejšom type pamäti sa do pamäti neukladajú iba samotné jedince vyberané z populácie v rôznych generáciách, ale s každým jedincom je vložená aj informácia o aktuálnom stave prostredia, v ktorom vyberaný jedinec bol v momente svojho výberu. Takáto pamäťová schéma je označovaná ako *asociatívna pamäť* [46].

V prípade použitia binárnej reprezentácie sa ako informácia o stave prostredia používa distribučný vektor – informácia o stave prostredia má tvar vektora reálnych čísel, pričom dĺžka daného vektora je rovná počtu pozícií použitých pre reprezentáciu tela jedincov v populácii. Hodnota nejakej pozície tohto vektora potom udáva pravdepodobnosť výskytu hodnoty 1 na príslušnej pozícii v aktuálnej generácii.

V prípade asociatívnej pamäti sa mení spôsob použitia informácie, ktorá je uchovávaná v pamäti. V prípade detekcie zmeny funkcie vhodnosti sa z pamäti vyberie najlepší záznam (jedinec má najlepšiu vhodnosť podľa aktuálnej funkcie vhodnosti) a s ním asociovaný pravdepodobnostný vektor sa použije na vygenerovanie skupiny jedincov, ktoré nahradia najhorších jedincov v populácii.

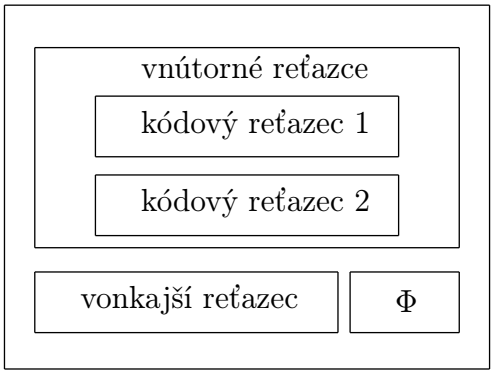
3.2.2 Algoritmy s implicitnou pamäťou

Implicitná pamäť poskytuje priestor pre pamätanie informácií, avšak nedáva k dispozícii prostriedky, ktoré by evolučnému algoritmu umožnili vkladať informácie do a vyberať ich z pamäti explicitným spôsobom[6]. Typickým pred-

[6]Nikde v štruktúre algoritmu nie je operácia "vloženie informácie do pamäti" ako ani operácia "výber informácie z pamäti".

staviteľom implicitnej pamäte sú prístupy využívajúce redundantnú formu kódovania v štruktúre jedincov.

Štandardný evolučný algoritmus kóduje riešenie úlohy v štruktúre jedinca tak, že informácia o každom prvku riešenia je kódovaná iba raz – používa sa teda neredundantné kódovanie. Pri redundantnom kódovaní sa informácia o každom prvku riešenia v štruktúre jedinca kóduje viackrát. Najrozšírenejším spôsobom redundantného kódovania je dvojnásobné kódovanie v usporiadaní podľa obr. 3.1, označované ako diploidná forma.



Obr. 3.1: Štruktúra jedinca pri použití diploidnej reprezentácie.

Pri diploidnej reprezentácii je jedinec kódovaný nie iba jedným reťazcom, ale namiesto toho dvomi reťazcami. Pre každú pozíciu tak existujú dve alternatívne hodnoty, každá v jednom z reťazcov. Dve hodnoty, prislúchajúce nejakej pozícii, však nemusia byť vždy rovnaké. Vzniká tak dodatočný problém, ktorá z týchto hodnôt (a či vôbec jedna z nich) má byť reprezentovaná navonok. Preto k použitým dvom kódovacím reťazcom, označovaným ako *vnútorné*, sa pridáva ďalší reťazec, označovaný ako *vonkajší*. Jeho úlohou je reprezentovať kódovanú informáciu navonok ako výsledný prejav daného jedinca.

Vonkajší reťazec slúži pre reprezentáciu jedinca, vlastne reprezentuje aktuálnu podobu riešenia, ktorého nositeľom je daný jedinec. Preto aj celková vhodnosť jedinca sa odvádza z obsahu vonkajšieho reťazca. Naopak, keďže obsah vonkajšieho reťazca je odvádzaný z obsahu oboch vnútorných reťazcov, procesy tvorby nových jedincov (aplikácia genetických operátorov) vonkajší reťazec ignorujú a stavajú výlučne na obsahu vnútorných reťazcov. Potomok od svojich rodičov preberá iba vnútorné reťazce (ktoré sú vhodne

upravené[7]) a vonkajší reťazec si vytvorí sám nezávisle od obsahu vonkajších reťazcov rodičov.

Redundantná pamäť, tvorená vnútornými reťazcami, umožňuje pamätať si viac informácií (o hodnotách zložiek a ich kombináciách, nájdených v minulosti) ako je obsiahnutých v obsahu vonkajšieho reťazca. Pre odvádzanie vonkajšieho reťazca z obsahov vnútorných reťazcov sa používa jedna z dvoch odvodzovacích schém:

- dominantno-recesívna schéma,
- aditívna schéma.

3.2.2.1 Dominantno-recesívne diploidy

Pri tejto schéme hodnota, vyskytujúca sa vo vonkajšom reťazci na nejakej pozícii, bude preberaná z niektorého z vnútorných reťazcov z danej pozície. Pokiaľ oba vnútorné reťazce na danej pozícii obsahujú rovnakú hodnotu, tak potom táto hodnota je prenesená aj do vonkajšieho vyjadrenia jedinca. V prípade, že vnútorné reťazce obsahujú rôzne hodnoty, je potrebné pre vonkajší reťazec zvoliť jednu z nich. Táto schéma sa snaží zabrániť tomu, aby hodnoty, ktoré aktuálne nie sú preferované, boli z populácie úplne vytlačené. Tieto hodnoty sa môžu vyskytovať v jednom z vnútorných reťazcov, pričom aktuálne nie sú prenášané do vonkajšieho vyjadrenia jedinca.

Pre uľahčenie voľby výberu hodnôt do vonkajšieho reťazca, jednotlivé hodnoty, ktoré sa môžu vyskytnúť v kóde jedinca, nie sú si navzájom rovno-

[7]Najprv sa pomocou krížiaceho operátora z vnútorných reťazcov rodičov vytvoria vnútorné reťazce potomkov, ktoré sú následne upravené mutačným operátorom. Oba operátory sú pritom použité iba s určitou pravdepodobnosťou. Keďže krížiaci operátor je možno použiť viacerými spôsobmi, je viacero spôsobov generovania nových jedincov, líšiacich sa rozsahom a spôsobom kombinovania materiálu z vnútorných reťazcov rodičov:

1) Vnútorné reťazce prvého rodiča sa skrížia navzájom. To isté sa vykoná aj pre druhého rodiča. Potomok si (náhodne) vyberie po jednom skríženom vnútornom reťazci od každého z rodičov. Druhý potomok je zostavený zo zvyšných skrížených vnútorných reťazcov rodičov. Každý potomok môže obsahovať materiál z oboch vnútorných reťazcov oboch rodičov.

2) Rodičia si navzájom vymenia jeden (náhodne vybratý) vnútorný reťazec, čím vzniknú dvaja potomkovia. Pre každého z potomkov sa následne jeho vnútorné reťazce navzájom skrížia. Každý potomok bude takto obsahovať materiál z jedného vnútorného reťazca každého z rodičov.

3) Krížia sa dvojice vnútorných reťazcov, pričom vždy jeden z dvojice pochádza od jedného rodiča a druhý z dvojice zase od druhého rodiča. Skrížené reťazce z dvojice sú následne rozdelené, každý z dvojice je vložený do iného z potomkov. Párovanie dvojíc vnútorných reťazcov rodičov je náhodné. Každý z potomkov môže obsahovať materiál zo všetkých vnútorných reťazcov oboch rodičov.

cenné. Hodnoty majú priradenú novú vlastnosť – dominanciu. Tie hodnoty, ktoré sú považované za dominantné, sú preferované oproti hodnotám, ktoré nie sú dominantné. Ak sa stretnú dve rôzne hodnoty s rôznou dominanciou, tak vždy je vyberaná tá z nich, ktorá je považovaná za dominantnú voči druhej hodnote.

V minulosti sa objavili pokusy o samoadaptívne určovanie dominancie jednotlivých hodnôt pre každú z pozícií kódovacích reťazcov osobitne – vnútorné reťazce nekódovali iba hodnoty prvkov riešenia, ale aj dominanciu týchto hodnôt. Zrejmou nevýhodou týchto prístupov bolo prílišné zväčšenie priestoru prehľadávania. Jednoduchší prístup nevyžadoval dominantnosť osobitne reprezentovať – bola určená pevným predpisom, platným pre všetky hodnoty na všetkých pozíciách. Starším reprezentantom pevne zadanej dominancie bola Holsteinova schéma. Niektoré novšie schémy rozširovali pôvodnú myšlienku s cieľom odstrániť jej neduhy. Príkladom môže byť schéma podľa [34].

Vonkajší reťazec používa binárnu reprezentáciu – v jeho tele sa môžu vyskytnúť iba dve hodnoty 0 a 1. Naproti tomu vnútorné reťazce používajú štyri možné hodnoty: 0_d – dominantná nula, 1_d – dominantná jednotka, 1_r – recesívna jednotka, 0_r – recesívna nula. Hodnoty 1_d a 1_r sa navonok prejavujú ako 1 a analogicky hodnoty 0_d a 0_r sa navonok prejavujú ako 0.

Prevod obsahu vnútorných reťazcov na vonkajší sa deje podľa týchto pravidiel:

- schéma je založená na princípe dominancie, pri stretnutí dominantnej a recesívnej hodnoty sa navonok prejaví vždy dominantná hodnota,
- pri stretnutí dvoch rovnakých dominantných hodnôt sa daná hodnota prejaví navonok; to isté platí aj pri stretnutí dvoch rovnakých recesívnych hodnôt,
- pri stretnutí dvoch rôznych dominantných hodnôt sa ľubovoľná z nich zmení na recesívnu (a navonok sa prejaví druhá hodnota, ktorá ostáva dominantná),
- pri stretnutí dvoch rôznych recesívnych hodnôt sa ľubovoľná z nich zmení na dominantnú (a prejaví sa navonok).

Aplikácia týchto pravidiel vedie na transformáciu hodnôt, znázornenú mapou dominancie v tab. 3.1.

Pre ilustráciu podstaty dominantno-recesívneho prístupu je možné použiť jednoduchý spojitý pravdepodobnostný model, inšpirovaný modelom

	0_d	0_r	1_r	1_d
0_d	0	0	0	0/1
0_r	0	0	0/1	1
1_r	0	0/1	1	1
1_d	0/1	1	1	1

Tabuľka 3.1: Mapa dominancie.

uvedeným v [34]. Pre jednoduchosť nech všetky reťazce v štruktúre jedincov kódujú iba jednu pozíciu.

Vnútorné reťazce sú náhodne inicializované, pričom každá možná hodnota je generovaná s rovnakou pravdepodobnosťou 0.25. Z pravdepodobností výskytu jednotlivých hodnôt $p(0_d)$, $p(0_r)$, $p(1_d)$ a $p(1_r)$ je možné určiť pravdepodobnosti výskytu všetkých prípustných párov hodnôt ako

$$\begin{array}{rcl} p(0_d0_d) & = & p(0_d)^2 \\ p(0_d0_r) & = & 2p(0_d)p(0_r) \\ p(0_r0_r) & = & p(0_r)^2 \\ p(0_d1_r) & = & 2p(0_d)p(1_r) + p(0_r)p(1_r) + p(0_d)p(1_d) \\ p(1_d0_r) & = & 2p(1_d)p(0_r) + p(0_r)p(1_r) + p(0_d)p(1_d) \\ p(1_r1_r) & = & p(1_r)^2 \\ p(1_d1_r) & = & 2p(1_d)p(1_r) \\ p(1_d1_d) & = & p(1_d)^2 \end{array} \tag{3.5}$$

Z pravdepodobností kombinácií hodnôt vo vnútorných reťazcoch je možné určiť pravdepodobnosti výskytu hodnôt vo vonkajších reťazcoch priamočiarym spôsobom podľa

$$\begin{array}{rcl} p(0) & = & p(0_d0_d) + p(0_d0_r) + p(0_r0_r) + p(0_d1_r) \\ p(1) & = & p(1_d1_d) + p(1_d1_r) + p(1_r1_r) + p(1_d0_r) \end{array} \tag{3.6}$$

Budeme predpokladať, že jedinec s vonkajším prejavom 0 má vhodnosť Φ, zatiaľ čo vonkajšiemu prejavu 1 zodpovedá vhodnosť $K\Phi$. Ak pre konštantu K platí $K > 1$, tak hodnota 1 bude preferovaná pred hodnotou 0, v opačnom prípade $K < 1$ tomu bude naopak.

Ak v úlohe selekcie použijeme proporcionálnu selekciu, tak pravdepodobnosť výskytu každého prípustného páru hodnôt po selekcii bude určená vzťahom analogickým k jednému zo vzťahov

$$p(0_d0_d) = \frac{p(0_d0_d)}{4(1+K)L} \qquad p(1_d1_d) = \frac{Kp(1_d1_d)}{4(1+K)L} \tag{3.7}$$

pričom všetky kombinácie s vonkajším prejavom 0 používajú podobný vzťah ako kombinácia 0_d0_d a všetky kombinácie s prejavom 1 zase vzťah ako kombinácia 1_d1_d. Hodnota L je normalizačným členom

$$L = \frac{1}{4(1+K)} \quad \begin{array}{c}[Kp(1_d1_d) + Kp(1_d1_r) + Kp(1_r1_r) + Kp(1_d0_r) + \\ p(0_d0_d) + p(0_d0_r) + p(0_r0_r) + p(0_d1_r)]\end{array} \tag{3.8}$$

Po selekcii rodičov sa vytvoria potomkovia, pričom sa použije iba kríženie. Distribúcia hodnôt vo vnútorných reťazcoch sa zmenila na

$$\begin{array}{rcl} p(0_d) & = & p(0_d0_d) + 0.5p(0_d0_r) + 0.5p(0_d1_r) \\ p(0_r) & = & p(0_r0_r) + 0.5p(0_d0_r) + 0.5p(1_d0_r) \\ p(1_r) & = & p(1_r1_r) + 0.5p(1_d1_r) + 0.5p(0_d1_r) \\ p(1_d) & = & p(1_d1_d) + 0.5p(1_d1_r) + 0.5p(1_d0_r) \end{array} \tag{3.9}$$

Následne je možné opätovne určiť pravdepodobnosti výskytu jednotlivých párov podľa (3.5) a pravdepodobnosti vonkajšieho prejavu podľa (3.6).

Podobný model je možné vytvoriť aj pre podobu algoritmu s neredundantnou reprezentáciou iba jedným reťazcom. V tomto prípade je model oveľa jednoduchší, pretože existujú iba dve možné hodnoty, pričom pravdepodobnosti ich výskytu sú $p(0)$ a $p(1)$. Na začiatku pri inicializácii sú obe rovné 0.5. Pri uvažovaní proporcionálnej selekcie sa menia podľa vzťahov

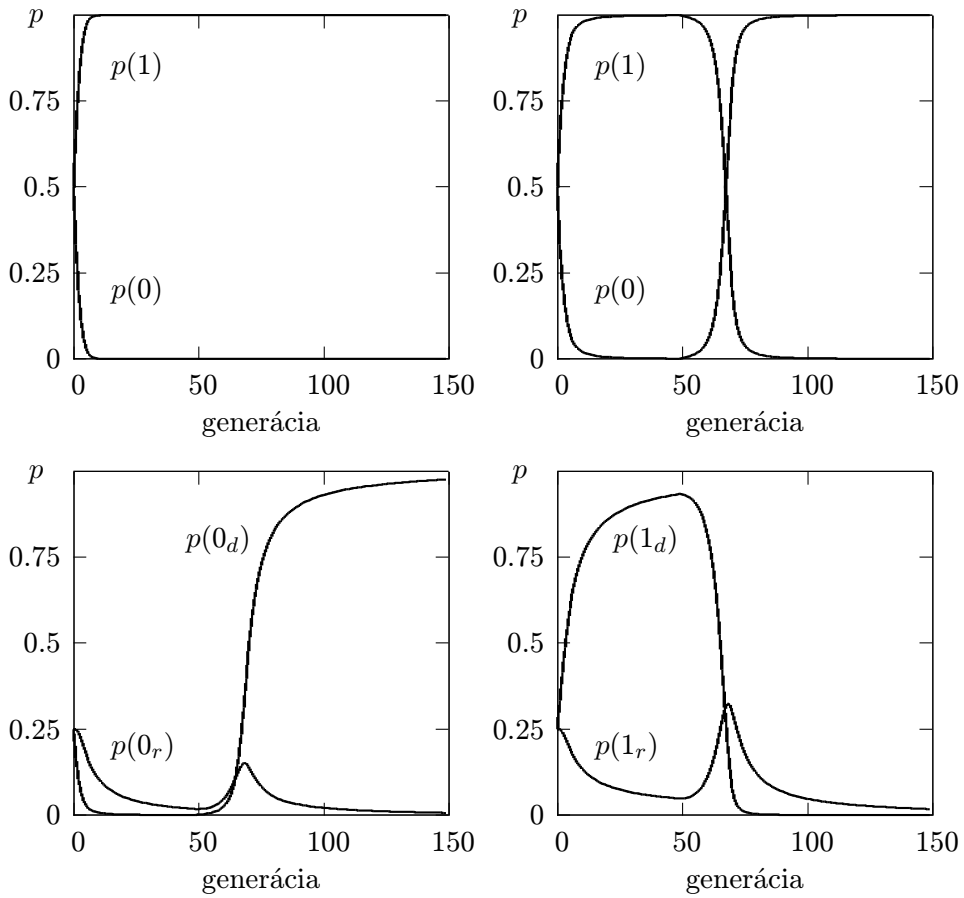
$$p(0) = \frac{p(0)}{(1+K)L} \qquad p(1) = \frac{Kp(1)}{(1+K)L} \tag{3.10}$$

pričom hodnota L je opäť normalizačným členom, teraz daným pomocou

$$L = \frac{1}{1+K}[Kp(1) + p(0)] \tag{3.11}$$

Obidva modely umožňujú pravdepodobnostne modelovať priebeh evolučného hľadania. Budeme simulovať vývoj[8] počas 150 generácií, pričom na začiatku bude preferovaná hodnota prejavu 1 (nastavením $K = 2$), avšak v generácii 50 dôjde k zmene preferencie ($K = 0.5$) a začne byť preferovaná hodnota 0. Výsledky simulácie sú na obr. 3.2. Neredundantnej forme zodpovedá priebeh vľavo hore. Je zrejmé, že populácia veľmi rýchlo skonverguje. V populácii ostávajú iba preferované hodnoty, aktuálne nepreferované hodnoty sú z populácie úplne vylúčené. Následkom toho algoritmus nie je schopný zareagovať na zmenu vhodnosti a ostáva zviazaný s už neaktuálnym riešením. Pri praktickom použití sa preto algoritmus musí spoliehať na

[8] Pri simulácii boli pravdepodobnosti uvažované s presnosťou na 5 desatinných miest.



Obr. 3.2: Výsledky pravdepodobnostných modelov neredundantnej a redundantnej dominantno-recesívnej schémy.

operátor mutácie, aby opätovne získal do populácie hodnoty, ktoré z nej boli v minulosti vytlačené.

Naproti tomu diploidná forma (vonkajší prejav je znázornený grafom vpravo hore) síce konverguje o niečo pomalšie, ale vďaka implicitnej pamäti nedošlo k úplnému vylúčeniu nepreferovaných hodnôt, čo algoritmu umožnilo zareagovať na zmenu vhodnosti a po jej zmene nájsť nové riešenie. Dôvod pre to je zrejmý zo situácie vo vnútorných reťazcoch (dolné priebehy). Dominantná forma nepreferovanej hodnoty síce bola z populácie vytlačená, avšak nepreferovaná hodnota ostala v populácii v dostatočnom množstve v podobe recesívnej formy. Rôznorodosť hodnôt v populácii je väčšia než bola v prípade neredundantnej reprezentácie.

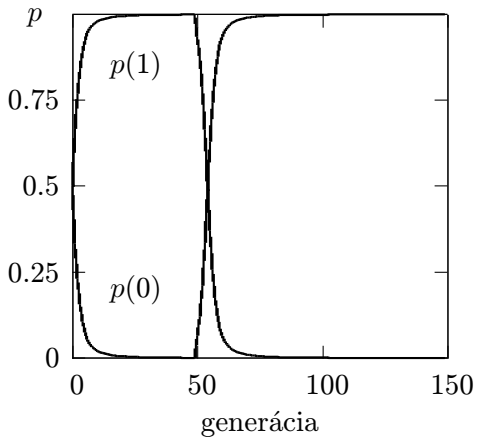
Dominantné hodnoty sú pri dominantno-recesívnych diploidných formách reprezentácie prenášané z vnútornej reprezentácie do vonkajšieho prejavu, zatiaľ čo recesívne hodnoty sú síce prítomné v štruktúre jedinca, ale väčšinou sa neprejavujú navonok. V prípade zmeny vhodnosti môže nastať presun polohy hľadaného riešenia do inej časti priestoru prehľadávania. To je situácia, keď je vhodné priamo aktivovať aj tie hodnoty, ktoré sú síce uchovávané v pamäti, avšak na prejave jedincov sa nepodieľajú.

K tomu stačí, aby sa zmenila dominancia hodnôt – aby sa recesívne hodnoty stali dominantnými (a dominantné sa tým pádom musia stať recesívnymi). Je to možné realizovať prekódovaním obsahu vnútorných reťazcov, kedy napríklad dvojica 1_d1_d sa vo vnútorných reťazcoch nahradí dvojicou 1_r1_r a naopak, dvojica 0_d0_d sa nahradí dvojicou 0_r0_r a naopak, a 1_d0_r sa nahradí 1_r0_d a naopak. Výsledkom je, že pri polovici možných kombinácií hodnôt vnútorných reťazcov sa zmení hodnota prezentovaná navonok. Signálom, že je žiadúce uskutočniť túto zmenu, môže byť napríklad výrazná zmena vhodnosti jedincov v populácii.

Zaradenie takejto zmeny dominancie do nášho pravdepodobnostného modelu vedie na priebeh vonkajšieho prejavu, ktorý je daný na obr. 3.3. Je zrejmé, že reakcia algoritmu je teraz rýchlejšia – zmena začína okamžite bez oneskorenia, prechodový dej je kratší a skôr nastáva ustálenie. Vysvetlenie je viditeľné na obr. 3.2 vľavo dole. Vďaka zmene dominancie krivka po generácii 50 začne rásť z o niečo vyšších hodnôt ako v prípade absencie zmeny dominancie (v generácii 50 sa hodnoty oboch kriviek vymenia) a to má vplyv na urýchlenie celého procesu.

3.2.2.2 Aditívne diploidy

Táto schéma, na rozdiel od predchádzajúcej, nie je založená na priamom preberaní hodnôt z vnútorných reťazcov do reťazca vonkajšieho. Hodnoty



Obr. 3.3: Výsledky pravdepodobnostného modelu dominantno-recesívnej schémy so zmenou dominancie.

v oboch vnútorných reťazcoch, umiestnené na nejakej pozícii, prispievajú k určeniu výslednej hodnoty na zodpovedajúcej pozícii vonkajšieho reťazca.

V prípade, že na nejakej pozícii vnútorných reťazcov sa nachádzajú rôzne hodnoty, tak tieto vnútorné reťazce prispievajú rôznou mierou k určeniu vonkajšieho prejavu jedinca, ktorým môže byť hodnota odlišná od oboch prispievajúcich hodnôt. V opačnom prípade, ak oba vnútorné reťazce obsahujú na nejakej pozícii rovnaké hodnoty, tak oba tieto reťazce prispievajú rovnakou mierou k vonkajšiemu vyjadreniu danej pozície. To však neznamená, že vo vonkajšom vyjadrení bude rovnaká hodnota ako hodnota, ktorá sa vyskytla vo vnútorných reťazcoch. Príčinou je to, že vnútorné reťazce pre účely reprezentácie môžu používať celkom inú množinu hodnôt ako je množina hodnôt používaná vo vonkajších reťazcoch, pretože hodnoty z vnútorných reťazcov nie sú preberané do reťazca vonkajšieho.

Aj keď je vo vonkajších reťazcoch preferovaná iba jedna hodnota a ostatné hodnoty boli z vonkajších reťazcov vytlačené, nemusí to znamenať vytláčanie hodnôt aj z vnútorných reťazcov, pretože každá z hodnôt, použiteľných vo vnútorných reťazcoch, môže prispievať k preferovanej hodnote vo vonkajších reťazcoch. Konvergencia hodnôt vonkajšieho prejavu teda nemusí znamenať konvergenciu hodnôt vnútornej reprezentácie.

Určovanie vonkajšieho prejavu jedinca sa realizuje prahovacou schémou. Príspevky vnútorných reťazcov sa aditívne zložia. Keďže vnútorné reťazce využívajú konečnú množinu možných hodnôt, tak množina možných výsled-

kov, získaných zložením oboch príspevkov, je taktiež konečná. Táto množina je následne prahovaná pomocou jedného prahu na dve možné hodnoty[9]. V prípade párneho počtu možných výsledkov aditívneho zloženia má prahovanie deterministický tvar:

- ak výsledok zloženia je menší ako prah, potom vo vonkajšom reťazci bude hodnota 0,
- ak výsledok zloženia je väčší ako prah, potom vo vonkajšom reťazci bude hodnota 1.

zatiaľ čo pri nepárnom počte možných výsledkov zloženia nadobúda prahovanie čiastočne stochastický charakter:

- ak výsledok zloženia je menší ako prah, potom vo vonkajšom reťazci bude hodnota 0,
- ak výsledok zloženia je rovnaký ako prah, tak hodnota vonkajšieho prejavu sa vyberá náhodne s rovnakými pravdepodobnosťami pre obe hodnoty,
- ak výsledok zloženia je väčší ako prah, potom vo vonkajšom reťazci bude hodnota 1.

Pri tomto druhom spôsobe prahovania existuje akési pásmo neurčitosti, keď nie je možné výslednú hodnotu určiť deterministickým spôsobom.

Metóda aditívnych diploidov[10] bola navrhnutá v [37]. Používa štyri rôzne hodnoty v_A, v_B, v_C a v_D vo vnútorných reťazcoch a binárnu reprezentáciu pre vonkajšie reťazce. Prevod obsahu vnútorných reťazcov na vonkajší sa deje podľa týchto pravidiel:

- pre skladanie príspevkov vnútorných reťazcov sa hodnoty 2, 3, 7 a 9 považujú za vyjadrenie hodnôt v_A, v_B, v_C a v_D (teda $v_A = 2$ a $v_D = 9$),
- pre prahovanie sa používa prah $\theta = 10.5$.

Aplikácia týchto pravidiel vedie na deterministickú[11] transformáciu hodnôt, znázornenú prahovacou mapou v tab. 3.2.

[9] Prípad, keď vonkajší reťazec používa binárnu reprezentáciu. V prípade viacárnej reprezentácie počet nutných prahov adekvátne narastá.

[10] Podobný pamäťový princíp bol použitý v [38] pri redundantnom kódovaní, keď jedinec síce používa iba jeden reťazec, avšak každá hodnota je redundantne kódovaná na niekoľkých susedných pozíciách tohto reťazca (schéma $Shades^2$ používa dve susedné hodnoty a $Shades^3$ zase tri susedné hodnoty).

[11] Pre nedeterministickú transformáciu stačí uvažovať napríklad hodnoty 0, 1, 1, 2 ako vyjadrenia hodnôt v_A, v_B, v_C a v_D.

	v_A	v_B	v_C	v_D
v_A	0	0	0	1
v_B	0	0	0	1
v_C	0	0	1	1
v_D	1	1	1	1

Tabuľka 3.2: Mapa prahovania.

Podstatou aditívnych diploidov je schopnosť pamätať si hodnoty, ktoré nie sú vo vonkajšom prejave, akosi skryte "v pozadí". Pre ilustráciu tejto vlastnosti uvažujme dvojicu hodnôt v_A a v_C. Vonkajším prejavom takejto dvojice je hodnota 0. Ak sa stretnú dvaja rodičia, z ktorých obaja majú vo vnútorných reťazcoch túto kombináciu, tak obaja rodičia majú taktiež rovnaký vonkajší prejav. Nový jedinec, vytváraný z týchto rodičov, zdedí vo svojich vnútorných reťazcoch jednu z kombinácií v_Av_A, v_Av_C, v_Cv_A alebo v_Cv_C, kde prvé hodnoty pochádzajú z prvého rodiča a druhé z rodiča druhého. A podľa uvedenej mapy prahovania je zrejmé, že v prípade zdedenia kombinácie v_Cv_C bude jeho vonkajší prejav opačný ako prejav oboch jeho rodičov. Aj keď teda z vonkajšieho prejavu nejaká hodnota je vytláčaná, pri tvorbe nových jedincov je s pravdepodobnosťou 0.25 spätne vkladaná do vonkajšieho prejavu novo vytváraných jedincov.

Schopnosť takejto pamäte majú však iba štyri dvojice v_Av_C, v_Bv_C, v_Av_D a v_Bv_D, zatiaľ čo ostatné dvojice túto schopnosť nemajú. Preto pre zachovanie tejto schopnosti sa zavádza nový operátor nazývaný "nútená mutácia", ktorý dvojicu bez danej schopnosti mení s nejakou pravdepodobnosťou na jednu z dvoch dvojíc s danou schopnosťou a rovnakým vonkajším prejavom ako menená dvojica. Príkladom takejto zmeny je $v_Av_B \rightarrow v_Av_C$ alebo $v_Av_B \rightarrow v_Cv_B$.

Podobne ako pre dominantno-recesívny prístup, aj teraz je možné vytvoriť jednoduchý spojitý pravdepodobnostný model, keď všetky reťazce v štruktúre jedincov kódujú iba jednu pozíciu. Vnútorné reťazce sú náhodne inicializované, pričom každá možná hodnota je generovaná s rovnakou pravdepodobnosťou 0.25. Z pravdepodobností výskytu jednotlivých hodnôt $p(v_A)$, $p(v_B)$, $p(v_C)$ a $p(v_D)$ je možné určiť pravdepodobnosti výskytu všetkých prípustných párov hodnôt ako

$$p(v_Av_A) = p(v_A)^2 \qquad p(v_Av_B) = 2p(v_A)p(v_B) \tag{3.12}$$

kde podľa prvého vzťahu sa určujú všetky homogénne páry (páry vytvorené z rovnakých hodnôt) a podľa druhého zase všetky nehomogénne páry

(kombinujúce rôzne hodnoty).

Z pravdepodobností kombinácií hodnôt vo vnútorných reťazcoch je možné teraz určiť pravdepodobnosti výskytu hodnôt vo vonkajších reťazcoch podľa

$$\begin{array}{rcl} p(0) & = & p(v_Av_A)+p(v_Av_B)+p(v_Bv_B)+p(v_Av_C)+p(v_Bv_C) \\ p(1) & = & p(v_Av_D)+p(v_Bv_D)+p(v_Cv_C)+p(v_Cv_D)+p(v_Dv_D) \end{array} \tag{3.13}$$

Opäť budeme predpokladať, že jedinec s vonkajším prejavom 0 má vhodnosť Φ, zatiaľ čo vonkajšiemu prejavu 1 zodpovedá vhodnosť $K\Phi$ (pri $K > 1$ hodnota 1 bude preferovaná pred hodnotou 0, v opačnom prípade $K < 1$ tomu bude naopak).

Ak teraz v úlohe selekcie použijeme proporcionálnu selekciu, tak pravdepodobnosť výskytu každého prípustného páru hodnôt po selekcii bude určená jedným zo vzťahov

$$p(v_Av_A) = \frac{p(v_Av_A)}{5(1+K)L} \qquad p(v_Dv_D) = \frac{Kp(v_Dv_D)}{5(1+K)L} \tag{3.14}$$

pričom všetky kombinácie s vonkajším prejavom 0 používajú rovnaký vzťah ako kombinácia v_Av_A a všetky kombinácie s prejavom 1 zase vzťah ako kombinácia v_Dv_D. Normalizačný člen L je teraz

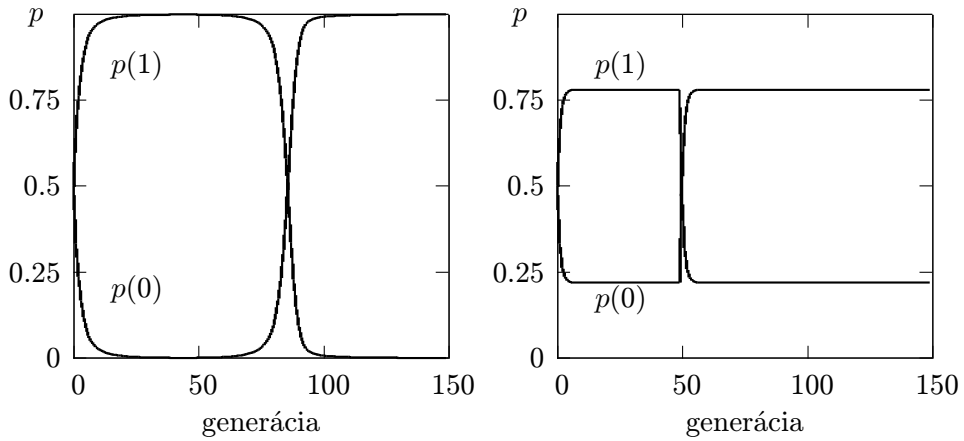
$$\begin{array}{rl} L = \frac{1}{5(1+K)} & [Kp(v_Av_D)+Kp(v_Bv_D)+Kp(v_Cv_C)+ \\ & \quad Kp(v_Cv_D)+Kp(v_Dv_D)+p(v_Av_A)+ \\ & \quad p(v_Av_B)+p(v_Bv_B)+p(v_Av_C)+p(v_Bv_C)] \end{array} \tag{3.15}$$

Po selekcii rodičov sa vytvoria potomkovia, pričom sa použije iba kríženie. Distribúcia hodnôt vo vnútorných reťazcoch sa teraz zmenila na

$$\begin{array}{rcl} p(v_A) & = & p(v_Av_A)+0.5p(v_Av_B)+0.5p(v_Av_C)+0.5p(v_Av_D) \\ p(v_B) & = & 0.5p(v_Av_B)+p(v_Bv_B)+0.5p(v_Bv_C)+0.5p(v_Bv_D) \\ p(v_C) & = & 0.5p(v_Av_C)+0.5p(v_Bv_C)+p(v_Cv_C)+0.5p(v_Cv_D) \\ p(v_D) & = & 0.5p(v_Av_D)+0.5p(v_Bv_D)+0.5p(v_Cv_D)+p(v_Dv_D) \end{array} \tag{3.16}$$

A opäť je možné opakovane určiť pravdepodobnosti výskytu jednotlivých párov podľa (3.12) a pravdepodobnosti vonkajšieho prejavu podľa (3.13).

Výsledky simulácie (rovnaké podmienky ako pri dominantno-recesívnych diploidoch) sú na obr. 3.4. Vľavo je prípad, keď operátor nútenej mutácie nie je použitý. Je zrejmé, že reakcia na zmenu preferencie hodnôt je veľmi oneskorená – pomalšia ako pri dominantno-recesívnej schéme (obr. 3.2 vpravo hore).



Obr. 3.4: Výsledky pravdepodobnostného modelu aditívnej schémy bez použitia (vľavo) a s použitím operátora nútenej mutácie (vpravo).

Použitie operátora nútenej mutácie je modelované nasledovnými zmenami pravdepodobností výskytu jednotlivých kombinácií:

$$\begin{array}{l} p(v_A v_C) = p(v_A v_C) + 0.5p(v_A v_A) + 0.5p(v_A v_B) + 0.5p(v_B v_B) \\ p(v_B v_C) = p(v_B v_C) + 0.5p(v_A v_A) + 0.5p(v_A v_B) + 0.5p(v_B v_B) \\ p(v_A v_D) = p(v_A v_D) + 0.5p(v_C v_C) + 0.5p(v_C v_D) + 0.5p(v_D v_D) \\ p(v_B v_D) = p(v_B v_D) + 0.5p(v_C v_C) + 0.5p(v_C v_D) + 0.5p(v_D v_D) \\ p(v_A v_A) = p(v_A v_B) = p(v_B v_B) = 0 \\ p(v_C v_C) = p(v_C v_D) = p(v_D v_D) = 0 \end{array} \tag{3.17}$$

Zaradenie tohto operátora (vpravo na obrázku) má dramatický vplyv – ako na rýchlosť reakcie a rýchlosť prechodového deja, tak aj na konvergenciu hodnôt vo vonkajšom prejave jedincov.

O niečo všeobecnejší prístup sa snaží [45]. Rovnako ako v predchádzajúcom prípade, vonkajší reťazec používa binárnu reprezentáciu. Vnútorné reťazce však majú možnosť použiť ľubovoľný počet hodnôt $v_1, \ldots, v_n$. Prevod hodnôt vnútorných reťazcov na jeden z dvoch možných vonkajších prejavov je realizovaný prahovaním, pričom za vyjadrenie hodnoty v_j sa považuje j (teda jej index). Ak sa stretnú dve hodnoty v_i a v_j, tak prahovanie je možné podľa

$$prah(v_i, v_j) = \begin{cases} 0, & i + j < n + 1 \\ 1, & i + j > n + 1 \end{cases} \tag{3.18}$$

V prípade rovnosti oboch súčtov je výsledkom hodnota 0 alebo 1, pričom výber je náhodný s rovnakými pravdepodobnosťami pre obe možnosti. Ak počet hodnôt, použitých vo vnútorných reťazcoch, je deliteľný štyrmi, tak prahovanie môže mať deterministický tvar podľa

$$prah(v_i, v_j) = \begin{cases} 0, & i+j < n+1 \\ 1, & i+j > n+1 \\ 0, & i+j = n+1, min(u,v) > n/4 \\ 1, & i+j = n+1, min(u,v) \leq n/4 \end{cases} \tag{3.19}$$

Dominantno-recesívna schéma v prípade zmeny vhodnosti, signalizujúcej presun polohy hľadaného riešenia do inej časti priestoru prehľadávania, realizovala zmenu dominancie. Analógia takejto zmeny bola navrhnutá aj pre aditívnu schému [22]. Táto analógia je založená na pojmoch povýšenia a degradácie:

- degradácia hodnoty v_i znamená nahradenie tejto hodnoty najbližšou nižšou hodnotou v_{i-1} (ale iba v prípade, že $i > 1$),
- povýšenie hodnoty v_i znamená nahradenie tejto hodnoty najbližšou vyššou hodnotou v_{i+1} (ale iba v prípade, že $i < n$).

Zmena sa rovnako ako v predchádzajúcom prístupe realizuje prekódovaním obsahu vnútorných reťazcov. Ak odpovedajúca hodnota vo vonkajšom reťazci je 1, tak hodnota vo vnútornom reťazci je degradovaná. V opačnom prípade, ak vo vonkajšom reťazci sa nachádza 0, tak hodnota vo vnútornom reťazci je povýšená. Mení sa hodnota iba v jednom vnútornom reťazci, pričom pre každú pozíciu sa náhodne určuje, v ktorom reťazci bude hodnota zmenená.

3.3 Porovnanie metód pre riešenie nestacionárnych úloh

Empirické porovnanie niektorých vybraných metód pre riešenie nestacionárnych úloh bolo založené na variácii úlohy z [19]. Jedná sa o skladanie hodnôt opakovaného použitia jednorozmernej testovacej funkcie – jedinec, ktorý je interpretovaný ako bod v k-rozmernom priestore, má vhodnosť určovanú podľa

$$\Phi(\vec{x}) = \sum_{i=1}^{k} f(x_i) \tag{3.20}$$

s postupným vyhodnocovaním f pre hodnotu každej súradnice jedinca, kde funkcia f je vytvorená pomocou sady pomocných funkcií

$$f(x) = \max_i g_i(x) \tag{3.21}$$

pričom každá z funkcií g_i má známy zvonovitý tvar daný vzťahom

$$g_i(x) = u_i(t)\ exp\left(-(v_i(t) - x)^2 / 2\sigma_i^2(t)\right) \tag{3.22}$$

ktorý je definovaný trojicou hodnôt u_i, v_i a σ_i. Tieto hodnoty sa môžu meniť medzi jednotlivými evolučnými cyklami, čím dochádza k časovej zmene celkovej funkcie vhodnosti.

Pri experimentoch bol použitý 4-rozmerný priestor ($k = 4$). V definícii funkcie f bolo použitých 20 pomocných funkcií g_i, pričom parametre týchto funkcií boli generované náhodne. Použilo sa 6 sád hodnôt týchto parametrov, čím sa získalo 6 rozličných tvarov funkcie f. Tieto tvary sú ilustrované na obr. 3.5. Je zrejmé, že testovací problém je konštruovaný takým spôsobom, aby globálny extrém mal vhodnosť $\Phi = k * 100 = 400$.

Uvedené tvary boli použité pre simulovanie dvoch typov časových zmien funkcie vhodnosti:

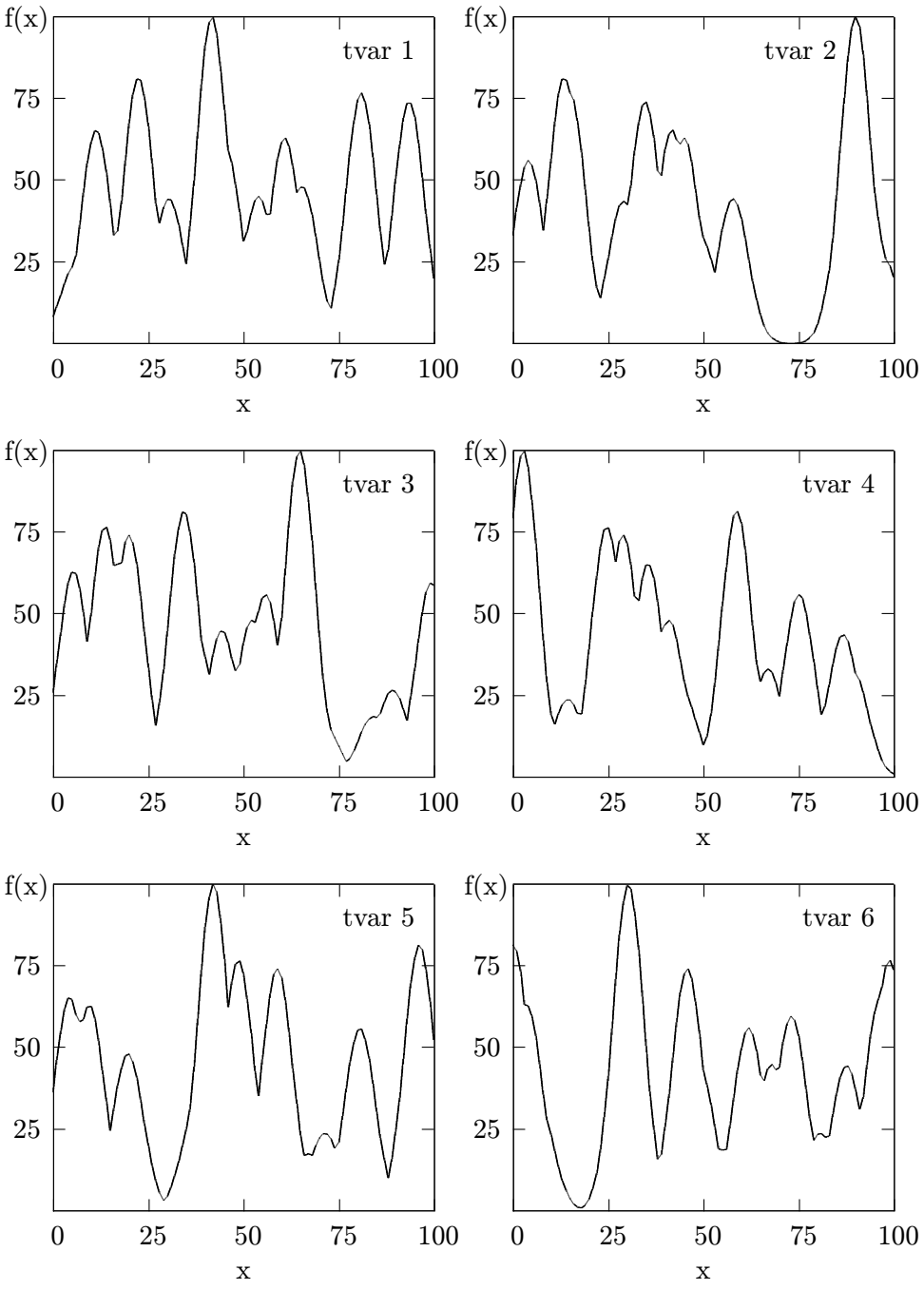
skokové zmeny – boli použité všetky tvary, pričom sledovaná doba bola rozdelená na šesť rovnakých úsekov a v každom z nich mala funkcia f jeden z tvarov (v i-tom úseku bol použitý i-ty tvar),

oscilačné zmeny – boli použité iba prvé dva tvary, ktoré sa periodicky prepínali.

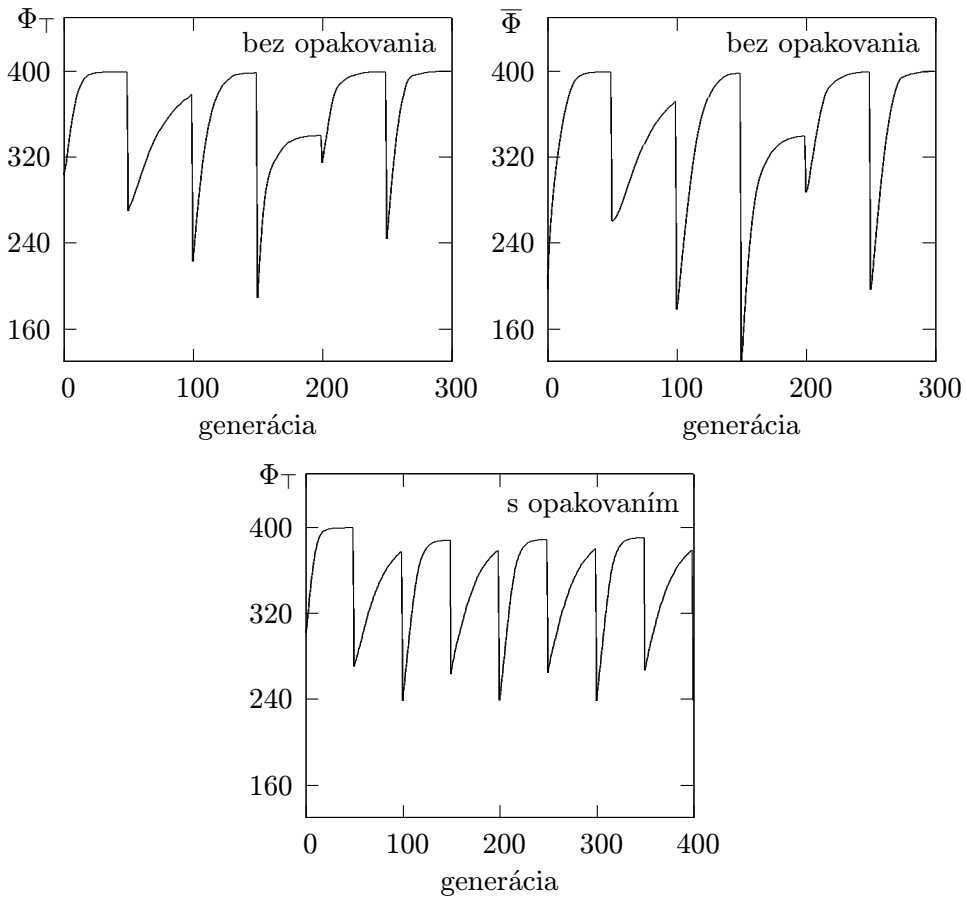
Je zrejmé, že poloha hľadaného extrému sa každou zmenou mení – najmenšia zmena bola pri prechode z piateho na šiesty tvar a najväčšia zase pri prechode z tretieho tvaru na štvrtý.

Ako testovacie prostredie bol použitý jednoduchý tvar testovacieho algoritmu, ktorý využíval nasledujúce prvky:

- Grayov binárny kód pre reprezentáciu hodnôt premenných (každá premenná bola reprezentovaná pomocou 12 bitov), celkovo boli použité 4 premenné,
- náhodnú inicializáciu populácie (o veľkosti 100) jedincov,
- triviálnu selekciu 100 rodičov, keď každý jedinec populácie sa stáva rodičom,



Obr. 3.5: Tvary funkcie f pre rôzne hodnoty u_i, v_i a σ_i.



Obr. 3.6: Výsledky pre testovací algoritmus.

- tvorbu potomkov dvojbodovým krížením ($p_c = 0.5$) a mutáciou zmeny bitu ($p_m = 1/(12 \cdot 4)$),
- deterministickú plus stratégiu – náhradu populácie najlepšími jedincami zo zjednotenia pôvodných jedincov populácie a vytvorených potomkov.

Štruktúra algoritmu umožňovala tento základný tvar rozširovať o rôzne metódy pre sledovanie polohy riešenia pri dynamických zmenách funkcie vhodnosti.

Chovanie použitého testovacieho algoritmu (bez rozšírení) je znázornené na obr. 3.6. Sledoval sa priebeh priemernej ($\overline{\Phi}$) a maximálnej ($\Phi_\top$) vhodnosti

v populácii, pričom sa skúmalo použitie všetkých šiestich tvarov funkcie bez opakovania jednotlivých tvarov (simulácia skokových zmien) a opakovanie prvých dvoch tvarov (simulácia oscilačných zmien). Tieto priebehy vyjadrujú priemerné hodnoty – bolo vykonaných 200 opakovaní a dosiahnuté výsledky boli spriemernené.

Priebehy korešpondujú s predpokladmi o obtiažnosti sledovania jednotlivých zmien – skutočne v prípade bez opakovania prechod z piateho na šiesty tvar nepredstavoval žiadny vážny problém, naproti tomu prechod z tretieho tvaru na štvrtý bol zvládnutý najhoršie. V prípade s opakovaním je zrejmé, že prechody nie sú symetrické – prechod z druhého tvaru na prvý nečiní zvláštny problém, zatiaľ čo o opačnom prechode to povedať nemožno. Príčina je zrejmá pri pohľade na prvé dva tvary na obr. 3.5. Riešenie, ktoré je globálnym v prvom tvare, má sklon uviaznuť ako lokálne riešenie v tvare druhom, pretože pre presun na pozíciu nového globálneho extrému musí prekonať výrazné údolie s veľmi malými vhodnosťami.

Otázkou je, ako by mali vyzerať uvedené priebehy v prípade použitia "dobrej" metódy pre riešenie nestacionárnych úloh. Tie zmeny by mali byť dvojakého charakteru:

- po každej zmene tvaru funkcie vhodnosti by sa priebeh maximálnej vhodnosti mal opäť vrátiť na hodnotu okolo 400 (teda by mala byť nájdená nová poloha hľadaného riešenia),
- doba prechodového deja (okamžitý pokles vhodnosti s jej následným stúpaním na pôvodnú hodnotu) by mala byť čo najkratšia.

Naopak, hĺbka prepadu vhodnosti po zmene tvaru funkcie vhodnosti nie je dôležitá (aj keď môže byť korelovaná s dobou prechodového deja). Rozdiel priebehov maximálnej a priemernej vhodnosti indikuje, do akej miery algoritmus udržuje rôznorodosť materiálu v populácii.

Do použitého testovacieho algoritmu boli ako jeho rozšírenia pridávané tieto metódy pre riešenie nestacionárnych úloh:

Náhodní imigranti. V každej generácii sa časť populácie nahradí náhodne generovanými jedincami. Nahrádza sa 15% celkového rozsahu populácie, pričom sa vždy deterministicky nahradia najhoršie jedince populácie.

Termodynamická náhrada. Termodynamický algoritmus v jeho statickej podobe – teplota bola nastavená exogénnym spôsobom a počas evolučného vývoja ostávala nemenná (bola použitá hodnota $T = 10$). Pri výbere jedincov do $P(t + 1)$ bol použitý výber bez náhrady – ak nejaký jedinec

bol vybraný za člena nasledujúcej generácie, tak bol z výberovej skupiny odstránený, čím sa zabránilo možnosti jeho opakovaného výberu.

Priama pamäť. Použila sa pamäť veľkosti 70% veľkosti samotnej populácie. Jej inicializácia bola náhodná. Pri aktualizácii pamäti sa najlepší jedinec populácie vyberal ako kandidát pre uchovanie v pamäti. Ak sa v pamäti nachádzali ešte náhodné jedince, ktoré ju inicializovali, tak vybraný jedinec nahradil jedného z nich. Ak všetky jedince v pamäti už boli uchovávanými jedincami z predchádzajúcich generácií, tak k danému vybranému jedincovi sa určil ten jedinec z pamäti, ktorý mu bol najpodobnejší. Ak vybraný jedinec z populácie mal lepšiu vhodnosť ako jedinec v pamäti, tak došlo k náhrade. V opačnom prípade k náhrade neprichádzalo a pamäť sa nemenila.

Signálom pre použitie obsahu pamäti bol pokles vhodnosti o viac ako 10%, pričom sa sledovala vhodnosť najlepšieho jedinca v populácii. V prípade výskytu takejto udalosti sa populácia nahradila najlepšími jedincami zo zjednotenia obsahu populácie a pamäti. Samotná pamäť sa pri tomto nemenila.

Dominantno-recesívne diploidy. Vnútorne sa používajú štyri hodnoty, ktoré majú význam dominantnej alebo recesívnej nuly či jednotky. Navonok sa prejavujú iba ako dve hodnoty, pričom transformácia medzi vnútorným a vonkajším vyjadrením je realizovaná mapou dominancie podľa tab. 3.1.

V prípade zmeny funkcie vhodnosti dochádza k zmene dominancie – prekódovaniu reprezentačných hodnôt vo vnútorných reťazcoch, pričom hodnota 1_d sa mení na 1_r a naopak, a hodnota 0_d na hodnotu 0_r a naopak. Súčasne dochádza k prekódovaniu vonkajšieho prejavu tam, kde v dvojici vnútorných reťazcov sa mení kombinácia 1_d0_r na 1_r0_d alebo 1_r0_d na 1_d0_r. Signálom pre túto zmenu bol pokles vhodnosti o viac ako 10%, pričom sa sledovala priemerná vhodnosť jedincov v populácii.

Aditívne diploidy. Pre reprezentáciu sa vo vnútorných reťazcoch používajú tri rôzne hodnoty v_A, v_B a v_C, pričom pre účely skladania sa ako ich vyjadrenie v aditívnej schéme uvažujú hodnoty 0, 1 a 2 (teda $v_A = 0$ atď.). Operátor nútenej mutácie je používaný na pravdepodobnostnom princípe s pravdepodobnosťou $p_{fm} = 0.3$ (dochádza k zmene iba pri kombináciách v_Av_A a v_Cv_C vo vnútorných reťazcoch). Pre prevod týchto hodnôt na hodnoty vo vonkajších reťazcoch slúži čiastočne stochastické prahovanie aditívneho pôsobenia hodnôt vo vnútorných reťazcoch pomocou dvoch prahov $\theta_1 = 1.5$ a $\theta_2 = 2.5$.

V prípade zmeny funkcie vhodnosti dochádza k zmene dominancie – prekódovaniu reprezentačných hodnôt vo vnútorných reťazcoch prostredníctvom ich povýšenia alebo degradácie. Signálom pre túto zmenu bol pokles vhodnosti o viac ako 10%, pričom sa sledovala priemerná vhodnosť jedincov v populácii.

Diploidy s výberom. Z vnútorných reťazcov sa určitým spôsobom generuje niekoľko rôznych vonkajších reťazcov. Ako výsledný vonkajší reťazec sa použije jednoducho ten z nich, ktorý má najlepšiu vhodnosť.

V zhode s [35] vo vnútorných aj vonkajších reťazcoch sú použité rovnaké dve hodnoty 0 a 1. Generujú sa štyria kandidáti na vonkajší reťazec. Dvaja z nich sú totožní s jedným z vnútorných reťazcov (teda každý z vnútorných reťazcov je zároveň kandidátom na vonkajší reťazec). Tretí kandidát sa vytvára tak, že:

- na pozíciách, na ktorých sa vnútorné reťazce zhodujú, sa hodnota z vnútorných reťazcov preberá,
- na pozíciách, na ktorých si vnútorné reťazce protirečia, sa použije hodnota 0.

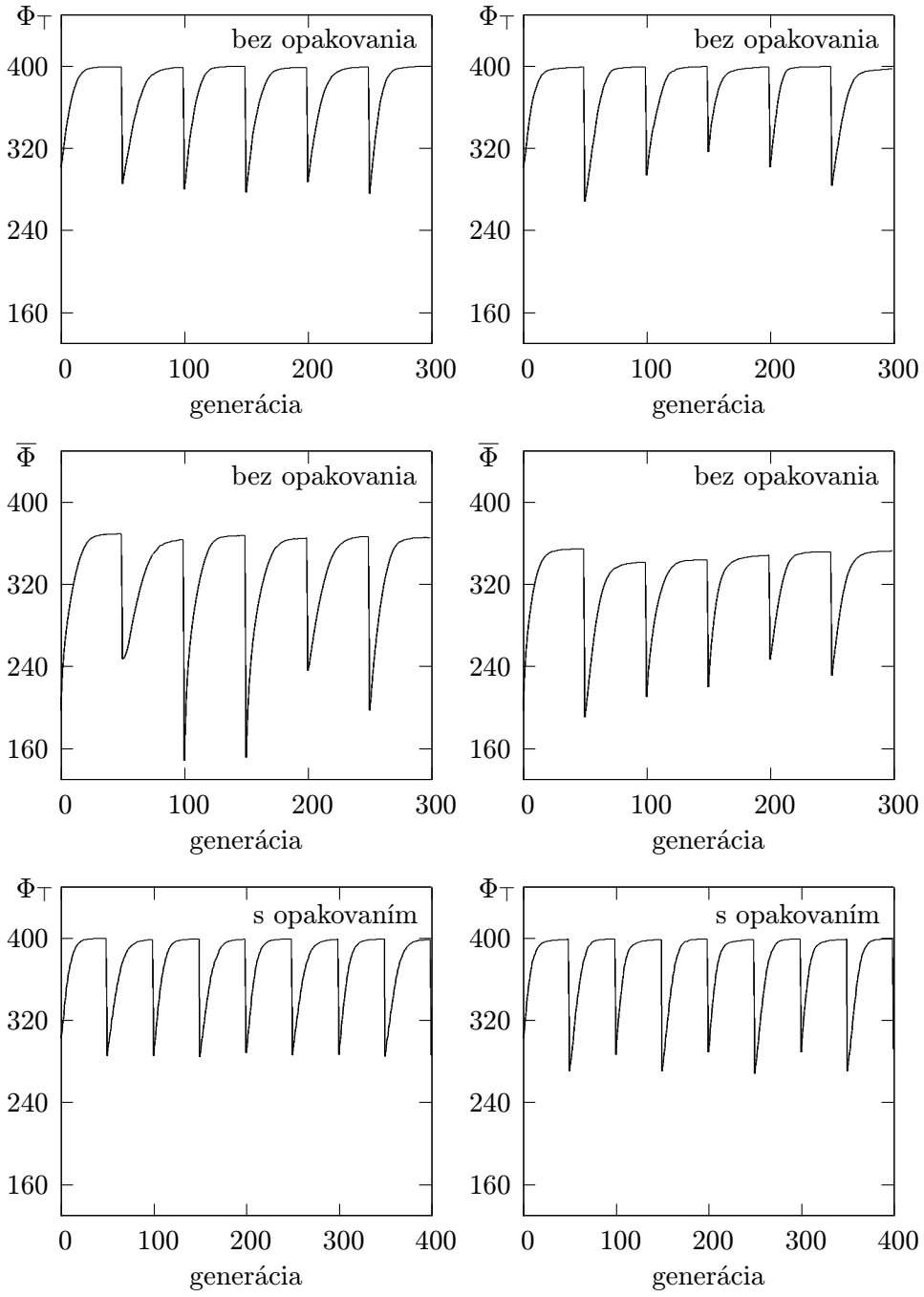
Štvrtý kandidát sa vytvára rovnakým spôsobom ako tretí iba s tým rozdielom, že na tých pozíciách, kde si vnútorné reťazce protirečia, sa použije hodnota 1.

Výsledky dosiahnuté týmito testovanými metódami sú prezentované na obr. 3.7 až obr. 3.9. Okrem diploidov s výberom, ostatné metódy vyžadovali nastavenie aspoň jedného parametra. Potrebné hodnoty boli nastavené tak, aby použité metódy vykazovali pre danú testovaciu funkciu čo najlepšie chovanie.

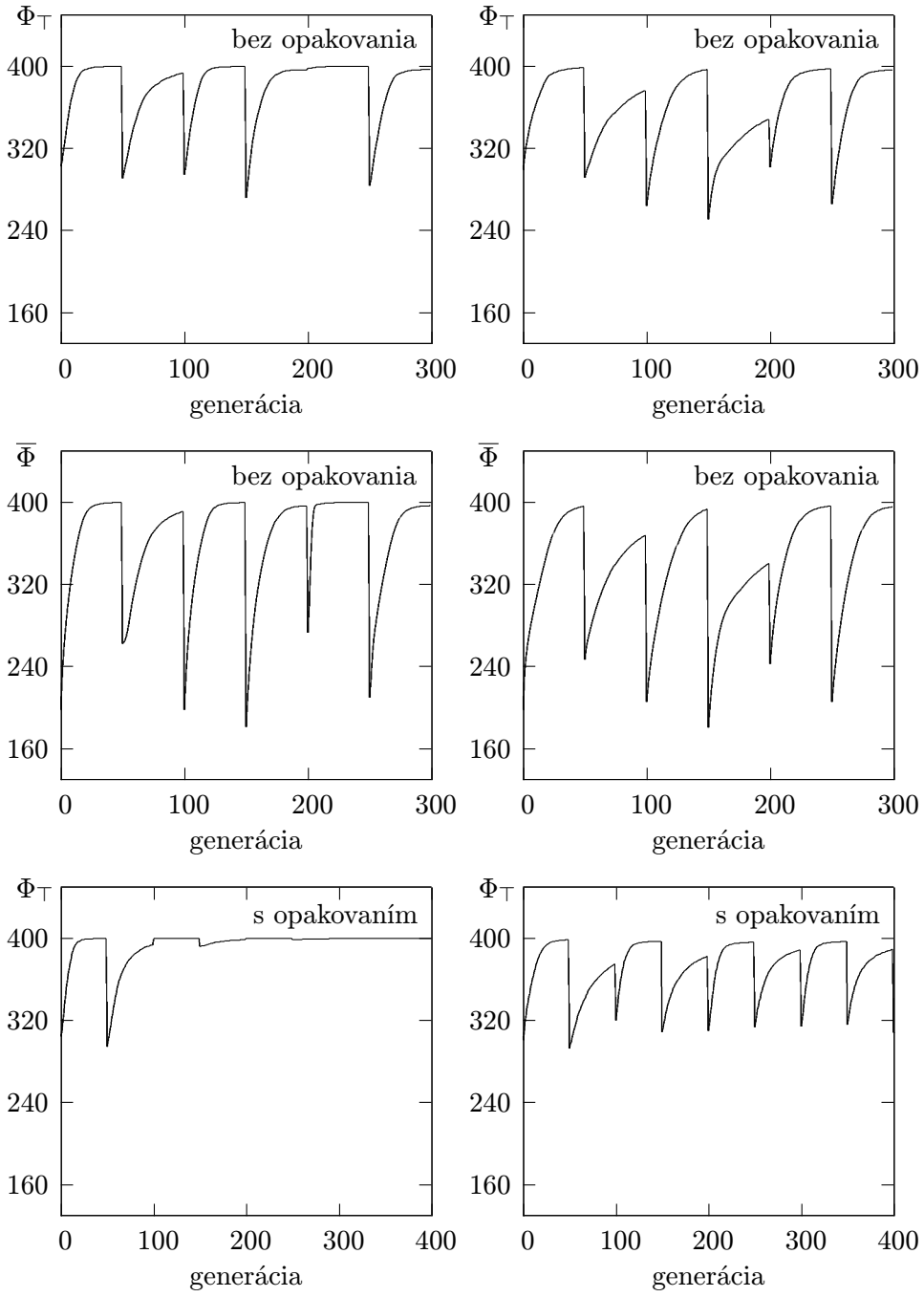
Z hľadiska udržiavania rôznorodosti populácie (signalizovaného rozdielom medzi hodnotami maximálnej a priemernej vhodnosti v prípadoch bez opakovania) je možné metódy rozdeliť do dvoch základných skupín. Rôznorodosť je významnejšie trvalo udržiavaná v prípade náhodných imigrantov, termodynamickej náhrady a aditívnych diploidov. V ostatných prípadoch možno pozorovať jasné prevládanie konvergencie priemernej vhodnosti k vhodnosti maximálnej[12].

Z hľadiska univerzálnosti použitia sa algoritmy s implicitnou pamäťou lepšie hodia pre prípad s oscilačným opakovaním tvarov funkcie vhodnosti.

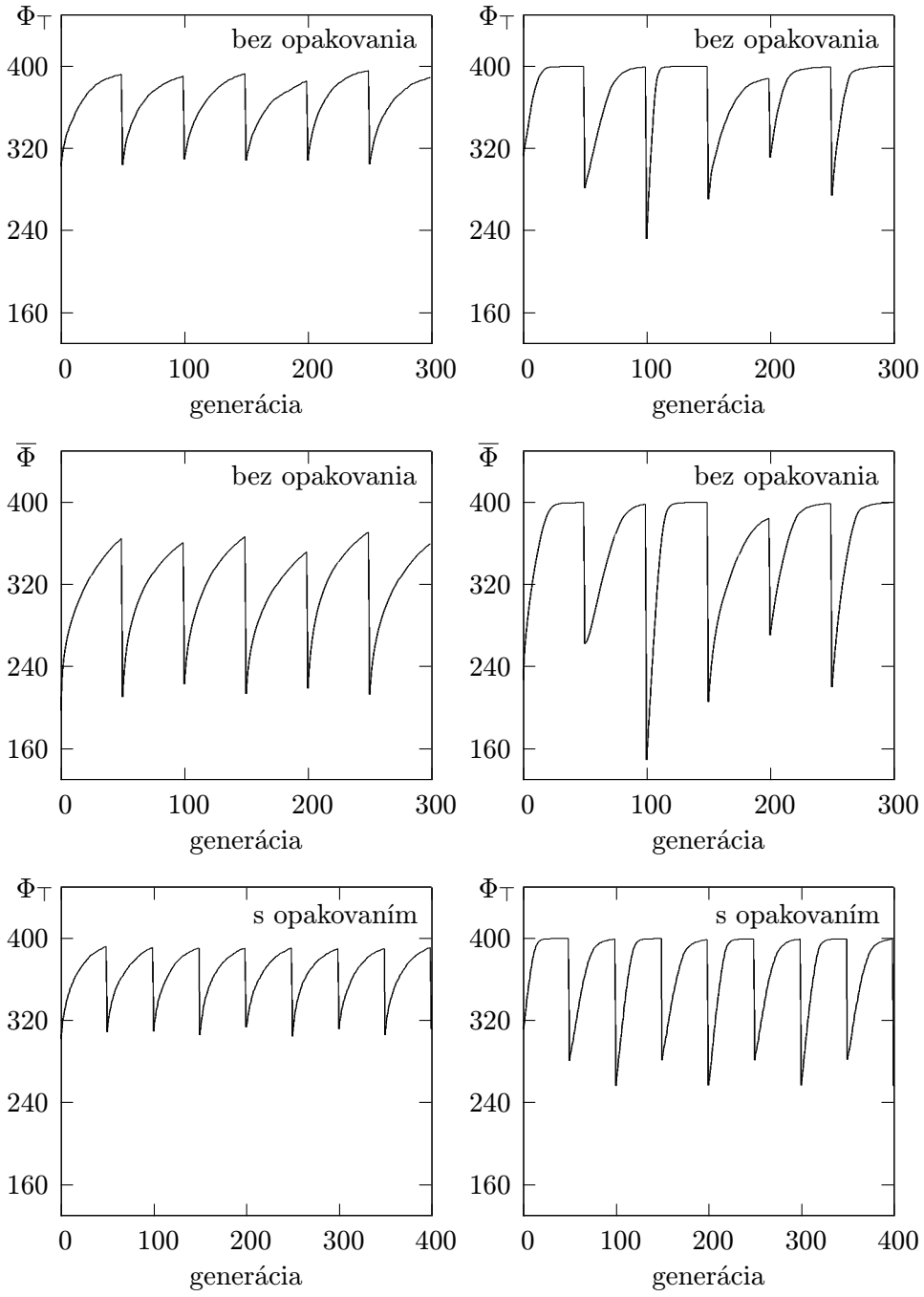
[12]Ostatné metódy tiež udržiavajú rôznorodosť, avšak buď mimo vonkajších prejavov jedincov alebo dokonca mimo samotnej aktuálnej populácie jedincov.



Obr. 3.7: Výsledky pre metódu náhodných imigrantov (vľavo) a termodynamickej náhrady (vpravo).



Obr. 3.8: Výsledky pre metódu priamej pamäti (vľavo) a dominantno-recesívnych diploidov (vpravo).



Obr. 3.9: Výsledky pre metódu aditívnych diploidov (vľavo) a diploidov s výberom (vpravo).

Pamäťové schopnosti diploidnej reprezentácie nie sú dostatočné pre prípad zmien, keď dochádza k výskytu stále nových polôh hľadaného riešenia a nedochádza k opakovaniu minulých situácií.

Aj keď sa na prvý pohľad zdá, že výkon metódy dominantno-recesívnych diploidov je aj v prípade oscilačných zmien degradovaný (pre druhý tvar testovacej funkcie), je tam jasný trend učenia sa a algoritmus sa s prechodom z prvého tvaru na druhý dokáže s postupujúcou skúsenosťou vysporiadavať stále lepšie a lepšie.

Najlepšie sa s oscilačnými zmenami dokázal vysporiadať algoritmus priamej pamäti, ktorý po počiatočnej fáze učenia (napĺňania pamäti vhodnými jedincami) dokáže sledovať polohu riešenia tak efektívne, akoby k žiadnym dynamickým zmenám funkcie vhodnosti ani nedochádzalo.

V prípade skokových zmien, pri ktorých nedochádza k opakovaniu, pamäťové prístupy prehrávajú s prístupmi zameranými na rôznorodosť (aj keď napríklad v prípade použitia priamej pamäti rozdiel nie je veľmi výrazný). Toto chovanie je očakávané, pretože keďže sa vyskytujú stále nové tvary funkcie vhodnosti, pamäť (či už implicitná alebo explicitná) nemala príležitosť naakumulovať vhodný materiál z minulej skúsenosti.

Niektoré metódy sú citlivejšie na to, aká je vzdialenosť a aký tvar má aktuálna funkcia vhodnosti medzi pôvodnou a novou polohou hľadaného extrému. Pri tých metódach, ktoré sú na toto citlivé, priebehy vhodnosti vyzerajú rôzne po rôznych zmenách – po nejakej zmene je rýchle nájdená nová poloha riešenia, inokedy zase toto hľadanie trvá dlhšiu dobu a niekedy nová poloha nemusí byť nájdená vôbec. Metódami s takýmto nepredikovateľným výsledkom sú priama pamäť, dominantno-recesívne diploidy a diploidy s výberom. Naproti tomu náhodní imigranti, termodynamická náhrada a aditívne diploidy vykazujú priebehy hľadania uniformného tvaru, rovnakého pre všetky tvary testovacej funkcie (aditívne diploidy sa však dokážu dostať iba do blízkosti riešenia bez jeho úplnej identifikácie).

Výkon niektorých metód vyzerá veľmi podobne. To však neznamená, že sa jedná o rovnocenné prístupy – z nejakého iného hľadiska môže byť rozdiel medzi nimi značne veľký. Príkladom je porovnávanie náhodných imigrantov a termodynamickej náhrady. Priebehy maximálnej vhodnosti vyzerajú veľmi podobne, avšak termodynamická náhrada je výpočtovo značne náročnejšia a teda v prípade náhodných imigrantov sa za rovnaký čas vykoná viac evolučných cyklov s následkom rýchlejšieho (v zmysle potrebného času a nie generácií) nájdenia novej polohy hľadaného riešenia.

Kapitola 4

Riešenie úloh s ohraničeniami

V predchádzajúcich častiach sa implicitne predpokladalo, že každý kandidát na riešenie, vytvorený ako nejaká kombinácia dostupných stavebných blokov, je hodný pozornosti – teda vytvorenia a ohodnotenia s prípadným výberom daného kandidáta. Pri takomto prístupe bolo potrebné skúmať každého z kandidátov, pretože hľadané riešenie sa mohlo principiálne vyskytovať v ľubovoľnej časti priestoru kandidátov a jeho prítomnosť sa nedala vylúčiť v žiadnom podpriestore tohto priestoru.

Mnohé definície úloh však explicitne stanovujú dodatočné podmienky, ktorých splnenie je nutné pre to, aby malo zmysel o danom kandidátovi uvažovať. Ak niektorý kandidát tieto podmienky nespĺňa, tak určite nie je prípustným riešením[1]. Takéto podmienky prípustnosti sa označujú ako *ohraničenia*.

Ohraničenia môžu mať rôzny tvar. V úlohách numerickej optimalizácie priestor kandidátov $\mathcal{M}_c$ je definovaný ako n-rozmerná kocka $\mathcal{M}_c \subseteq \Re^n$ a každý kandidát má tvar n-rozmerného vektora hodnôt $\vec{x}$. Pre takto definovaný priestor kandidátov sú naviac dané ohraničenia, ktoré môžu mať jednu z dvoch typických foriem:

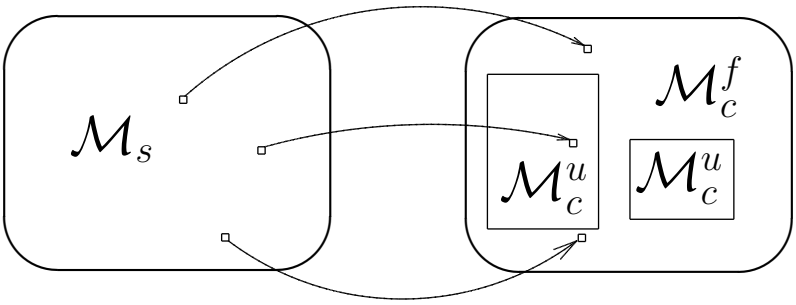
$$o_i(\vec{x}) \leq 0, i = 1, \ldots, k_1 \qquad o_i(\vec{x}) = 0, i = k_1 + 1, \ldots, k_1 + k_2 \tag{4.1}$$

V úlohách kombinatorickej optimalizácie je situácia pestrejšia, variácia rozličných typov ohraničení je väčšia. V zásade sa jedná o situácie, v ktorých sa zohľadňujú vzťahy medzi jednotlivými zložkami (čo sa dá s čím kombinovať),

[1] Na druhej strane splnenie týchto podmienok síce kandidátovi zabezpečuje prípustnosť ale nie optimálnosť.

vlastnosti použitých zložiek alebo hodnoty rôznych kritérií, definovaných nad použitými zložkami.

Prienik definovaných ohraničení a samotného priestoru kandidátov vyčleňuje prípustné oblasti. Takýmto spôsobom je priestor kandidátov delený na prípustnú $\mathcal{M}_c^f$ a neprípustnú $\mathcal{M}_c^u$ oblasť, pričom každá z nich môže byť tvorená jedným alebo viacerými navzájom dizjunktnými podpriestormi tak, ako to je znázornené v pravej časti obr. 4.1. Vo všeobecnosti nie je možné robiť žiadne predpoklady o tomto delení ohľadom počtu, veľkosti ani tvaru týchto podpriestorov.



Obr. 4.1: Priestor prehľadávania (vľavo) a priestor kandidátov (vpravo).

Ako bolo uvedené napríklad v [26], evolučný algoritmus nepracuje priamo s priestorom kandidátov. Jednotliví kandidáti sú reprezentovaní nejakým vhodným spôsobom a tieto ich reprezentácie vytvárajú priestor, s ktorým je algoritmus schopný narábať. Toto je priestor prehľadávania $\mathcal{M}_s$ (na obr. 4.1 znázornený vľavo), ktorý evolučný algoritmus prehľadáva a snaží sa v ňom identifikovať tie reprezentácie, ktoré reprezentujú hľadané riešenia.

Vzťah medzi týmito dvomi priestormi je riešený pomocou dekodéra g, ktorý dekóduje reprezentácie z $\mathcal{M}_s$ na kandidátov v $\mathcal{M}_c$. Pri úlohách s definovanými ohraničeniami tento dekodér môže byť v zásade dvojaký:

- dekóduje $\mathcal{M}_s$ na $\mathcal{M}_c$, teda každý kandidát môže byť výsledkom dekódovania,
- dekóduje $\mathcal{M}_s$ iba na $\mathcal{M}_c^f$, teda výsledkom dekódovania môže byť iba taký kandidát, ktorý spĺňa definované ohraničenia a teda sa nachádza v prípustnej oblasti.

V prvom prípade aj v priestore prehľadávania existujú prípustné a neprípustné oblasti, ktoré sú dekódované na prípustné a neprípustné oblasti priestoru kandidátov. Algoritmus musí byť schopný sa vysporiadať s výskytom

neprípustných kandidátov. V druhom prípade celý priestor prehľadávania je prípustný a algoritmus vôbec nevytvára neprípustných kandidátov.

Ďalším faktorom, ovplyvňujúcim riešenie takýchto úloh, je spôsob definovania funkcie vhodnosti, ohodnocujúcej kvalitu jednotlivých kandidátov. Vhodnosť môže byť definovaná

- na celom priestore kandidátov,
- iba pre kandidátov, nachádzajúcich sa v prípustnej oblasti.

V prvom prípade je možné použiť štandardný tvar evolučného algoritmu. Môže však nastať situácia, keď globálny extrém funkcie vhodnosti je umiestnený v zakázanej oblasti a použitie štandardného tvaru algoritmu by mohlo skončiť skonvergovaním do neprípustnej oblasti – pre zohľadnenie ohraničení je preto nutné zabrániť tejto konvergencii a umožniť algoritmu hľadať najlepšieho kandidáta z prípustných kandidátov.

Druhý prípad definície funkcie vhodnosti reprezentuje situáciu, keď kandidátov v zakázanej oblasti nie je možné porovnávať na základe vhodnosti. Takúto situáciu možno riešiť buď vhodným dodefinovaním funkcie vhodnosti aj pre neprípustných kandidátov alebo použiť dekódovanie, vyhýbajúce sa neprípustnej oblasti.

4.1 Hľadanie v prípustnej oblasti

Na problém s ohraničeniami je možný náhľad ako na viackriteriálny problém. Úlohu jedného kritéria hrá funkcia vhodnosti, úlohu ostatných kritérií hrajú jednotlivé ohraničenia (zvyčajne v podobe miery nesplnenia daných ohraničení)[2]. Pri tomto prístupe je vhodnosť uvažovaná oddelene od ohraničení. Snahou je nesplnenie ohraničení minimalizovať a súčasne vhodnosť maximalizovať alebo minimalizovať v závislosti na type riešeného optimalizačného problému. Rozdiel voči "pravým" viackriteriálnym úlohám je ten, že teraz nie všetky jedince z Paretovej množiny sú si navzájom rovnocenné. Jedince, ktoré spĺňajú ohraničenia, sú preferované voči jedincom, ktoré ohraničenia porušujú, nezávisle na hodnotách prvého kritéria – vhodnosti.

Aj keď je možné použiť viackriteriálny prístup, jeho použitie je pomerne zriedkavé – zvyčajne sa používajú iné prístupy. Najčastejšie používané prístupy pre riešenie úloh s ohraničeniami patria do jednej z týchto štyroch oblastí:

[2] Je možné každé ohraničenie uvažovať ako samostatné kritérium alebo všetky ohraničenia môžu byť kumulované do jedného spoločného kritéria.

- penalizačné funkcie,
- dekodéry,
- opravné procedúry,
- operátory zachovávajúce ohraničenia.

Úspešnosť týchto prístupov silne závisí na pomere veľkosti prípustnej oblasti k neprípustnej, tvare a umiestnení neprípustnej oblasti a definícii funkcie vhodnosti.

4.1.1 Penalizačné funkcie

Toto je najstarší a najrozšírenejší prístup k zohľadňovaniu existencie ohraničení. Vzťah medzi priestorom prehľadávania a priestorom kandidátov je daný zobrazením

$$g : \mathcal{M}_s \rightarrow \mathcal{M}_c \tag{4.2}$$

ktoré má rovnakú podobu ako v prípade evolučného algoritmu pre riešenie úloh bez ohraničení. Algoritmus prehľadáva celý priestor prehľadávania $\mathcal{M}_s$ bez akéhokoľvek obmedzenia a pracuje s kandidátmi z ľubovoľnej časti priestoru kandidátov $\mathcal{M}_c$ – s tými, ktoré spĺňajú všetky ohraničenia, ale aj s tými, ktoré ich porušujú.

Jediným zásahom do algoritmu je modifikácia vhodnosti jednotlivých kandidátov. Cieľom tejto úpravy je nasmerovať algoritmus k vytváraniu kandidátov z prípustnej oblasti tým, že sa zvýši pravdepodobnosť výberu jedincov, dekódovaných na kandidátov spĺňajúcich ohraničenia, na úkor zníženia pravdepodobnosti výberu jedincov, vedúcich na neprípustných kandidátov.

Splnenie alebo nesplnenie ohraničení sa premietne do výslednej vhodnosti – pôvodná vhodnosť je upravená pridaním *penalizačnej* funkcie. Jedinec a_i tak bude namiesto vhodnosti $\Phi(a_i)$ používať

$$\Phi(a_i) \quad + \quad w\ \Psi(a_i) \tag{4.3}$$

kde $\Psi(a_i)$ je penalizácia jedinca a_i, zohľadňujúca spĺňanie alebo porušovanie stanovených ohraničení daným jedincom[3].

Hodnota w je vyjadrením váhy penalizácie vzhľadom na hodnoty vhodnosti. Nastavenie tejto váhy je veľmi dôležité. Pri malých hodnotách je penalizácia slabá a hrozí nebezpečenstvo, že populácia skonverguje k neprípustnému riešeniu. Naopak, pri veľkých hodnotách vedúcich k silnej penalizácii

[3]Penalizácia môže zohľadňovať buď všetky ohraničenia alebo iba nejakú vybranú podmnožinu, bez uváženia ľahko splniteľných alebo obtiažne formalizovateľných ohraničení.

hrozí nebezpečie konvergencie síce v prípustnej oblasti, avšak mimo hľadaného riešenia (najmä v prípade, ak hľadané riešenie leží na hranici prípustnej oblasti). Intuitívne by penalizácia mala byť čo najmenšia, tesne nad limitom, keď optimálnym sa stáva neprípustné riešenie.

Aj keď zostavenie penalizačnej funkcie môže byť doménovo závislé, je možné použiť penalizačnú funkciu založenú na [42]:

- počte porušených ohraničení,
- miere porušenia ohraničení.

Prvý typ reprezentuje stupňovitú funkciu, ktorá môže obsahovať veľké ploché úseky (mnoho kandidátov porušuje rovnaký počet ohraničení), ktoré neposkytujú algoritmu dostatočné vodítko pre proces prehľadávania (vedúce na slabú diferenciáciu medzi neprípustnými jedincami).

Preto sa častejšie používa druhý typ, umožňujúci rozlišovať aj medzi jedincami, ktoré porušujú rovnaký počet ohraničení, avšak rôznym spôsobom. V tomto prípade má penalizácia tvar

$$\Psi \left(\; d(a_i, \mathcal{M}_c^f) \; \right) \tag{4.4}$$

monotónnej neklesajúcej funkcie založenej na vzdialenosti ohodnocovaného jedinca a_i od prípustnej oblasti. Keďže určenie tejto vzdialenosti nie je vždy jednoduché, tak sa často používa alternatíva, uvažujúca jednotlivé ohraničenia osobitne. Pre každé ohraničenie sa určuje, do akej miery ho daný jedinec porušuje, a jednotlivé porušenia sú následne navzájom kombinované do výslednej penalizácie.

Princíp penalizácie podľa vzdialenosti je možné ilustrovať na probléme obchodného cestujúceho, ktorý je typickým reprezentantom kombinatorickej optimalizácie. Pri tomto probléme je úlohou prejsť cez určitý počet miest v takom poradí, aby ohodnotenie výslednej cesty bolo čo najlepšie[4].

Nie každý prechod je však prípustným – za validné sú považované iba také prechody, ktoré každým mestom prejdú práve raz. Z takto definovanej požiadavky vyplývajú dve ohraničenia:

- počet navštívených miest musí byť rovnaký ako je celkový počet uvažovaných miest,
- nie je možné navštíviť žiadne mesto viac ako raz (a teda v zozname navštívených miest každá dvojica reprezentuje dve navzájom rôzne mestá).

[4] Ako kritérium ohodnotenia je možné použiť dĺžku cesty, avšak sú možné aj iné kritériá.

Splnenie prvého ohraničenia je jednoduché – napríklad stačí použiť reprezentáciu s pevnou dĺžkou, keď reprezentované budú iba také cesty, ktoré prechádzajú cez stanovený počet miest.

Druhé ohraničenie je obtiažnejšie – môže sa vyskytnúť cesta, ktorá síce prechádza stanoveným počtom miest, avšak niektoré mestá vynecháva a niektoré navštívi viackrát. Príkladom môže byť reprezentácia a_i vedúca na kandidáta $g(a_i)$ tvaru cesty

$$2 - 6 - 8 - 4 - 5 - 3 - 6 - 1 - 9 \tag{4.5}$$

ktorý síce prechádza cez deväť miest, ale zabudol navštíviť mesto 7. Týmto evidentne došlo k porušeniu stanoveného ohraničenia, otázkou však ostáva, do akej miery toto ohraničenie bolo porušené.

Pri úlohách kombinatorickej optimalizácie sa veľmi často vzdialenosť k prípustnej oblasti (odchýlka Δ) meria ako úsilie, potrebné na opravu kandidáta. Pre opravu cesty v ukážke je potrebné do nej doplniť mesto 7 a vyhodiť jednu návštevu mesta 6. V najjednoduchšom prípade jeden výskyt mesta 6 sa priamo nahradí mestom 7. V závislosti od toho, ktorý výskyt sa nahrádza, by miera porušenia ohraničenia mohla byť

$$\Delta(a_i) = \frac{d(2,7) + d(7,8)}{d(2,6) + d(6,8)} \qquad \Delta(a_i) = \frac{d(3,7) + d(7,1)}{d(3,6) + d(6,1)} \tag{4.6}$$

kde d je vzdialenosť medzi mestami. Prvý tvar zodpovedá náhrade prvého výskytu mesta 6 mestom 7 a druhý zase náhrade výskytu druhého. Použiť sa môže jedna z týchto hodnôt alebo ich priemer.

Pri úlohách numerickej optimalizácie je určenie miery porušenia numerického ohraničenia jednoduché. V prípade, že uvažované ohraničenie o_i má tvar nerovnosti $o_i(\vec{x}) \leq 0$, tak

$$\Delta_i(\vec{x}) = \begin{cases} o_i(\vec{x}), & o_i(\vec{x}) > 0 \\ 0, & inak \end{cases} \tag{4.7}$$

a v prípade, že má tvar rovnosti $o_i(\vec{x}) = 0$, tak

$$\Delta_i(\vec{x}) = \mid o_i(\vec{x}) \mid \tag{4.8}$$

Statická penalizácia. Tento spôsob penalizácie je založený na nemennosti váhových koeficientov – pomer uvažovania vhodností jedincov a ich penalizácií je konštantný počas celého procesu evolučného hľadania. Upravená vhodnosť má tvar [42]

$$\Phi(a_i) + w\ \Psi(a_i) = \Phi(a_i) + \sum_{j=1}^{n} w_j \left(\Delta_j(a_i)\right)^{\alpha} \tag{4.9}$$

kde parameter α umožní zohľadňovať veľkosti porušení ohraničení nelineárnym spôsobom[5]. Použitie osobitnej váhy pre každé ohraničenie umožňuje súčasne odlíšiť aj vplyv rôznych ohraničení a takýmto spôsobom nastaviť pomer, v ktorom porušenie rôznych ohraničení prispieva do výslednej penalizácie.

Nevýhodou tohto spôsobu penalizácie je nutnosť vhodne nastaviť niekedy pomerne veľký počet váhových koeficientov, čo zvyčajne vyžaduje iteračné nastavovanie pomocou početnej sady experimentov. Pre zjednodušenie tohto nastavenia váh je možné zredukovať počet váhových koeficientov (v extrémnom prípade na jeden) alebo použiť jednu z nasledujúcich penalizácií.

Bariérová penalizácia. Podstatou tohto spôsobu je dosiahnuť jasné oddelenie validných kandidátov od tých, ktorí porušujú ohraničenia. Cieľom je taká penalizácia, aby aj ten najhorší validný jedinec mal vhodnosť lepšiu ako najlepší nevalidný jedinec. Určovanie vhodnosti prebieha dvojkrokovo:

- určovanie penalizácie podľa rovnakého vzťahu ako pri statickej penalizácii (pre všetkých jedincov, validných aj nevalidných),
- úprava hodnôt nevalidných jedincov.

V druhom kroku sa upravujú hodnoty pre jedince, porušujúce ohraničenia, takým spôsobom, aby výsledná penalizovaná vhodnosť týchto jedincov bola horšia ako vhodnosť validných jedincov. Jednou z možností je podľa [29] určiť rozdiel medzi maximálnou vhodnosťou validných jedincov a minimálnou penalizovanou vhodnosťou jedincov, porušujúcich ohraničenia. O tento rozdiel sa následne zvýši penalizovaná vhodnosť všetkých nevalidných jedincov[6].

Pri tomto spôsobe nie je potrebné vážiť penalizáciu voči vhodnosti explicitnou váhou. A v prípade, že nie je potrebné rôzne ohraničenia uvažovať rôznym spôsobom, nie je potrebné používať žiadne váhové koeficienty.

Dynamická penalizácia. Nevýhodou statickej penalizácie je aj to, že v rôznych fázach hľadania nemôže plniť rozdielne funkcie. Použitie dynamických hodnôt umožní meniť silu penalizácie. Najčastejším cieľom je na

[5] Najčastejšie sa používajú hodnoty 1 a 2 vedúce na lineárne alebo nelineárne zohľadnenie miery porušenia ohraničení.

[6] Uvedený postup platí pri minimalizačnej úlohe, pre maximalizačnú úlohu musí byť upravený.

začiatku použiť slabšiu penalizáciu, aby algoritmus získal dostatočnú informáciu aj z prehľadávania tej časti priestoru prehľadávania, ktorá je dekódovaná na neprípustnú časť priestoru kandidátov. V neskorších fázach je potrebné zabezpečiť konvergenciu algoritmu k riešeniu, spĺňajúcemu všetky ohraničenia. Preto je vhodné použiť silnejšiu penalizáciu.

Pri tomto spôsobe penalizácie by sa teda váhový koeficient mal s narastajúcim časom postupne zvyšovať. Jednou z možností je sekvencia váh podľa [29]

$$w_{i+1} = \frac{1}{2\ T_{i+1}} \tag{4.10}$$

kde parameter T (označovaný ako teplota[7]) sa mení od nejakej počiatočnej hodnoty T_0 (často $T_0 = 1$) podľa predpisu

$$T_{i+1} = \alpha T_i \tag{4.11}$$

kde koeficient α (menší než 1) udáva rýchlosť tejto zmeny. Zmena môže nastať každú generáciu alebo po uplynutí určitého počtu generácií.

Adaptívna penalizácia. V predchádzajúcich spôsoboch penalizácie sa váhové koeficienty určovali bez ohľadu na aktuálnu situáciu procesu hľadania. Cieľom adaptívnej penalizácie je reagovať na aktuálnu situáciu a penalizáciu prispôsobovať tejto situácii.

Aktuálna situácia môže byť meraná rôznymi charakteristikami. V príklade podľa [14] je používaný pomer počtu validných a počtu nevalidných jedincov v aktuálnej populácii. Využíva sa vzťah medzi týmto pomerom a hodnotou váhových koeficientov:

- zmenšovanie penalizácie má za následok menšie potláčanie kandidátov porušujúcich ohraničenia, a teda ich počet v populácii stúpa,
- zvyšovanie penalizácie tlačí algoritmus do prípustnej oblasti a teda počet validných jedincov v populácii sa zvyšuje.

Pri využívaní takejto závislosti sa potom váhový koeficient mení podľa jedného z dvoch vzťahov

$$w_{t+1} = \alpha w_t \qquad\qquad w_{t+1} = w_t / \alpha \tag{4.12}$$

kde koeficient α (väčší než 1) udáva veľkosť jednotkovej zmeny. Násobenie (a zvyšovanie váhy) sa používa v prípade, ak v populácii je menšie percento

[7] Autor metódy sa očividne inšpiroval algoritmom simulovaného žíhania.

validných jedincov ako je požadované, a delenie (a teda znižovanie váhy) prichádza na rad v prípade, ak v populácii je percento validných jedincov vyššie oproti požadovanej hodnote.

Pretože hodnota váhového koeficienta sa adaptívne upravuje, nastavenie jeho počiatočnej hodnoty nie je kritické. Jedna z možností je

$$w_0 = \frac{\sum_{i=1}^{\mu} \Phi(a_i)}{\sum_{i=1}^{\mu} \sum_{j=1}^{n} \Delta_j(a_i)} \tag{4.13}$$

zrovnávajúca sumu vhodností v populácii s porušeniami všetkých n ohraničení všetkými μ jedincami.

4.1.2 Dekodéry

V rámci tohto prístupu dekodér je hlavným a zároveň jediným prvkom evolučného algoritmu, ktorý zohľadňuje existenciu ohraničení. Ostatné súčasti algoritmu môžu mať štandardnú podobu, známu z riešenia úloh bez ohraničení. Princípom je chápanie dekódovania reprezentácií na kandidátov ako zobrazenie

$$g : \mathcal{M}_s \to \mathcal{M}_c^f \tag{4.14}$$

ktoré každý prvok z priestoru prehľadávania $\mathcal{M}_s$ zobrazuje na taký prvok z priestoru kandidátov, ktorý sa nachádza v jeho prípustnej časti $\mathcal{M}_c^f$. Inak povedané, žiadny kandidát, ktorý porušuje aspoň jedno z definovaných ohraničení (a teda sa nachádza v neprípustnej časti priestoru kandidátov $\mathcal{M}_c^u$), nie je reprezentovaný v priestore prehľadávania. Následkom toho počas evolučného vývoja nedochádza ku generovaniu reprezentácií nevalidných kandidátov a evolučný algoritmus teda neustále pracuje iba s kandidátmi, ktorí spĺňajú všetky ohraničenia.

Okrem uvedenej vlastnosti by vhodne navrhnutý dekodér mal mať aj niekoľko ďalších vlastností:

- každý kandidát z prípustnej časti priestoru kandidátov by mal byť reprezentovaný v priestore prehľadávania,
- všetci kandidáti, ktorí sú reprezentovaní, by mali byť reprezentovaní rovnakým počtom rôznych reprezentácií,
- podobné reprezentácie by mali reprezentovať podobných kandidátov,
- transformácia g by nemala byť výpočtovo náročná.

Prvá z nich poskytuje šancu, že algoritmus môže nájsť hľadaného kandidáta bez ohľadu na to, v ktorej časti prípustnej oblasti $\mathcal{M}_c^f$ sa tento kandidát nachádza. Nedodržanie tejto podmienky úplnosti apriórne vylučuje niektorých kandidátov napriek tomu, že spĺňajú všetky stanovené ohraničenia[8].

Druhá vlastnosť zabezpečuje to, že samotné dekódovanie nevnáša preferovanie jedných kandidátov voči iným kandidátom. Nerovnomernosť počtu reprezentácií by spôsobovala to, že pravdepodobnosť dekódovania jedného kandidáta pri náhodnom výbere prvku z priestoru prehľadávania by bola vyššia pre toho jedinca, ktorému zodpovedá viac reprezentácií.

A tretia požiadavka vlastne presadzuje princíp lokality, pri ktorom malá zmena v priestore prehľadávania má za následok zmenu v priestore kandidátov, ktorá je tiež malá. Nedodržanie tohto princípu by mohlo spôsobovať situáciu, keď malým zmenám v reprezentácii zodpovedajú veľké zmeny kandidátov a teda riadené prehľadávanie priestoru prehľadávania vedie na náhodné prehľadávanie v priestore kandidátov (prehľadávaniu nejakej malej oblasti v $\mathcal{M}_s$ by zodpovedalo prehľadávanie celého $\mathcal{M}_c^f$).

Prístup založený na zohľadnení stanovených ohraničení vhodne navrhnutým dekodérom je považovaný za účinný a teda vhodný pozornosti, avšak má aj nevýhody:

- dekodér je doménovo špecifický, nie je možný návrh "všeobecného" dekodéru, použiteľného pre viaceré navzájom rôzne úlohy,
- zostrojenie dekodéra, ktorý by zohľadňoval všetky ohraničenia, nie je triviálnou úlohou – zložitosť tohto návrhu môže byť porovnateľná so zložitosťou pôvodného problému.

Pre ilustráciu princípu dekodéru v oblasti kombinatorickej optimalizácie možno použiť problém obchodného cestujúceho. Uvažujme kandidáta, ktorý má tvar '4–7–8–3–2–5–9–1–6', teda sa jedná o cestu, ktorá začína v meste '4', pokračuje cez mesto '7', následne cez iné mestá a záverečne končí v meste '6'. Takýto kandidát je prípustným, spĺňa obe ohraničenia. Jeden spôsob možnej reprezentácie je mnohoznaková reprezentácia, kedy jedinec je reprezentovaný vektorom s n pozíciami, pričom na každej pozícii môže byť jedna z n hodnôt v_1, v_2 až v_n. Pri takejto reprezentácii by daný kandidát bol reprezentovaný ako vektor hodnôt

$$v_4, v_7, v_8, v_3, v_2, v_5, v_9, v_1, v_6 \tag{4.15}$$

Je možné použiť dekodér jednoduchého tvaru – výskyt hodnoty v_i bude dekódovaný na návštevu mesta i a pozícia hodnoty v_i znamená, v ktorom kroku

[8] Ak je isté, že medzi vylúčenými kandidátmi sa nenachádza hľadané riešenie, potom takéto vylúčenie nie je na závadu.

cesty k tejto návšteve dôjde. Pri použití štandardných operátorov môže byť vygenerovaný jedinec tvaru (došlo k mutácii hodnôt na tretej a piatej pozícii)

$$v_4, v_7, v_4, v_3, v_1, v_5, v_9, v_1, v_6 \tag{4.16}$$

s ktorým si už uvedený jednoduchý dekodér nedokáže poradiť – jedinec je transformovaný na kandidáta '4–7–4–3–1–5–9–1–6', ktorý spĺňa už iba jedno z ohraničení a teda patrí do neprípustnej časti priestoru kandidátov. Niektoré reprezentácie sú teda daným dekodérom dekódované na kandidátov, ktorí spĺňajú všetky ohraničenia, zatiaľ čo iné reprezentácie sú dekódované na kandidátov porušujúcich ohraničenia.

Iný spôsob možnej reprezentácie je mnohoznaková reprezentácia, kedy jedinec je opäť reprezentovaný vektorom s n pozíciami, pričom však teraz na i-tej pozícii môže byť hodnota v_j pre $j \in \{1, \ldots, n+1-i\}$ (teda na prvej pozícii môže byť ľubovoľná hodnota, na druhej pozícii už nemôže byť v_n a na poslednej pozícii môže byť iba v_1). Pri takejto reprezentácii by daný kandidát bol reprezentovaný napríklad ako vektor hodnôt

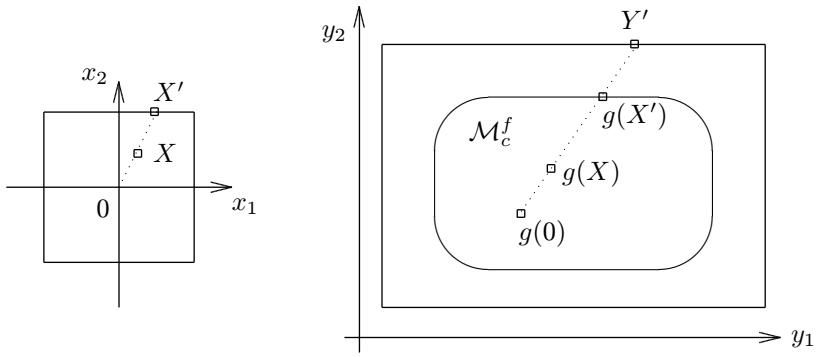
$$v_6, v_3, v_2, v_4, v_4, v_3, v_1, v_2, v_1 \tag{4.17}$$

kde hodnota v_i už nie je zviazaná s mestom i. Pre dekódovanie takto reprezentovaného jedinca je možné použiť dekodér využívajúci referenčný zoznam uvažovaných miest [29] – výskyt hodnoty v_i bude dekódovaný na návštevu mesta uvedeného na i-tej pozícii referenčného zoznamu a pozícia hodnoty v_i znamená, aký je aktuálny tvar referenčného zoznamu, ktorý sa skracuje s návštevou nejakého mesta práve o toto mesto[9]. Ako vidno, viacnásobný výskyt rovnakej hodnoty v reprezentácii nie je transformovaný na viacnásobnú návštevu toho istého mesta. Takéto dekódovanie každú reprezentáciu, ktorá sa môže vyskytnúť pri tomto spôsobe reprezentácie, transformuje na kandidáta, ktorý spĺňa všetky ohraničenia. Žiadna z reprezentácií nie je dekódovaná na kandidáta, porušujúceho nejaké z ohraničení.

Prístup založený na návrhu vhodného dekodéru je možné použiť aj v oblasti numerickej optimalizácie. Priestor kandidátov aj priestor prehľadávania majú rovnaký tvar – sú tvorené kartézskym súčinom domén jednotlivých

[9]V ukážke bol použitý referenčný zoznam (9 8 7 6 5 4 3 2 1). Hodnota v_6 odkazuje na šiestu pozíciu v zozname. Mesto '4', nachádzajúce sa na tejto pozícii, sa stáva prvým navštíveným mestom a keďže už bolo navštívené, tak je zo zoznamu odstránené. Po dekódovaní prvých troch pozícií bol zoznam skrátený na (9 6 5 3 2 1) a preto hodnota v_4 je dekódovaná na mesto '3' a zoznam je následne skrátený na (9 6 5 2 1). Druhá hodnota v_4 je potom dekódovaná na návštevu mesta 2.

premenných, pričom tieto domény obsahujú reálne hodnoty z nejakého intervalu. Jeden z možných spôsobov dekódovania je uvedený v [20]. Jeho použitie je ilustrované na obr. 4.2.



Obr. 4.2: Dekódovanie z priestoru prehľadávania (vľavo) do prípustnej časti priestoru kandidátov (vpravo).

Uvažujme najjednoduchší prípad, keď prípustná časť priestoru kandidátov $\mathcal{M}_c^f$ je tvorená jednou súvislou oblasťou, ktorá je súčasne aj konvexná[10]. Je zrejmé, že priestor prehľadávania $\mathcal{M}_s$ sa od $\mathcal{M}_c^f$ líši ako tvarom tak aj veľkosťou. Keďže táto odlišnosť bude stále prítomná, je možné priestor prehľadávania zvoliť ľubovoľne, bez ohľadu na tvar prípustnej oblasti priestoru kandidátov. Jednou z možností je symetrická voľba každej strany ako intervalu $< -1, 1 >$, vedúca na tvar mnohorozmernej kocky. Dekodér potom musí:

- stred priestoru prehľadávania (pri použitej voľbe jeho počiatok) zobraziť na bod vo vnútri prípustnej oblasti priestoru kandidátov,
- bod na hranici priestoru prehľadávania zobraziť na kandidáta na hranici prípustnej oblasti,
- bod z vnútra priestoru prehľadávania zobraziť na bod vo vnútri prípustnej oblasti.

Celé zobrazenie je realizované takým spôsobom, že ak reprezentácia leží na spojnici iných dvoch reprezentácií, tak táto reprezentácia sa zobrazí na kandidáta, ktorý leží na spojnici tých dvoch kandidátov, ktorí sú obrazmi tých

[10] Ak dva body patria do danej oblasti, potom každý bod na spojnici týchto dvoch bodov patrí do danej oblasti tiež.

reprezentácií, na spojnici ktorých ležala zobrazovaná reprezentácia. Navyše, ak zobrazovaná reprezentácia delí spojnicu, na ktorej leží, v určitom pomere, tak jej obraz rozdelí spojnicu, na ktorej leží, v tom istom pomere.

Inak povedané, ak je potrebné zobraziť nejakú reprezentáciu na kandidáta, je potrebné nájsť iné dve reprezentácie. Pre ne musí platiť nielen to, že zobrazovaná reprezentácia leží na ich spojnici, ale súčasne pre tieto dve reprezentácie musí byť známe, na akých kandidátov sa zobrazujú. Preto do úlohy týchto pomocných reprezentácií sa vyberá počiatok a priemet bodu na okraj priestoru. Ak zobrazovaná reprezentácia je bod v n-rozmernom priestore $X = [v_1, \ldots, v_n]$, potom jeho priemet na okraj priestoru sa určí ako

$$X' = \left[\frac{v_1}{max_{i=1}^{n} \mid v_i \mid}, \cdots, \frac{v_1}{max_{i=1}^{n} \mid v_i \mid} \right] \tag{4.18}$$

Reprezentácia $[0, \ldots, 0]$ reprezentuje stred priestoru prehľadávania. V ideálnom prípade by preto $g(0)$ mal byť stredom prípustnej oblasti priestoru kandidátov. Toto však nie je možné, keďže jej stred nie je známy. Preto úlohu $g(0)$ bude hrať ľubovoľný kandidát, spĺňajúci všetky ohraničenia[11].

Keďže tvar $\mathcal{M}_c^f$ vo všeobecnosti nie je známy, nie je možné priamo určiť $g(X')$. Preto je použitý nepriamy postup – najprv je určovaný kandidát Y' na okraji priestoru kandidátov a až následne sa určí $g(X')$. Poloha bodu Y' vzhľadom na $\mathcal{M}_c$ je rovnaká ako poloha bodu X' vzhľadom na $\mathcal{M}_s$. Ak $X' = [v'_1, \ldots, v'_n]$ a zároveň hodnota i-tej súradnice bodu Y' je z intervalu $< u_i^{min}, u_i^{max} >$, potom hodnota tejto i-tej súradnice bodu Y' sa určí ako

$$\frac{u_i^{max} + u_i^{min}}{2} + v'_i \, \frac{u_i^{max} - u_i^{min}}{2} \tag{4.19}$$

Pri známom bode Y' a bode $g(0)$ už nie je problém určiť bod $g(X')$. Určí sa ako ten bod na spojnici Y' a $g(0)$, ktorý leží na hranici prípustnej oblasti[12]. Po nájdení $g(X')$ a $g(0)$ už nie je problém určiť $g(X)$.

V prípade, že prípustná oblasť priestoru kandidátov nie je konvexná alebo nie je spojitá (a teda pozostáva z viacerých navzájom oddelených oblastí), tak je potrebné riešiť ešte jeden dodatočný problém – nie každý kandidát, ktorý leží na spojnici bodov $g(0)$ a $g(X')$, musí spĺňať všetky ohraničenia. Pre určovanie polohy $g(X)$ sa potom uvažujú iba tie časti spojnice medzi bodmi $g(0)$ a $g(X')$, ktoré ležia v prípustnej oblasti.

[11] V praktickej realizácii je jeden kandidát, spĺňajúci všetky ohraničenia, vyberaný náhodným spôsobom pri inicializácii evolučného algoritmu a následne plní túto úlohu počas celého evolučného vývoja.

[12] Pre jeho nájdenie je možné použiť napríklad metódu postupného polenia intervalov.

Existujú aj iné spôsoby dekódovania pre numerickú optimalizáciu. V [18] uvedený prístup je založený na vypĺňaní nepravidelnej prípustnej oblasti kružnicami v hexagonálnom usporiadaní a následnej zmene polomerov vpísaných kružníc tak, aby výsledný tvar bol aproximáciou jednotkovej kružnice. Jedná sa však o výpočtovo náročnejší prístup, ako prístup uvedený v predchádzajúcom texte.

4.1.3 Opravné procedúry

Pri tomto prístupe je evolučný algoritmus, ktorý môže mať štandardnú podobu známu z riešenia úloh bez ohraničení, doplnený o procedúru, umožňujúcu prácu s tými jedincami, ktoré porušujú niektoré z ohraničení. Samotné dekódovanie reprezentácií je chápané ako dvojkrokový proces, pozostávajúci z dvoch zobrazení

$$g_1 : \mathcal{M}_s \to \mathcal{M}_c^f \cup \mathcal{M}_c^u \qquad g_2 : \mathcal{M}_c^u \to \mathcal{M}_c^f \tag{4.20}$$

pričom prvé z nich každý prvok z priestoru prehľadávania $\mathcal{M}_s$ zobrazuje na prvok z priestoru kandidátov, či už v jeho prípustnej alebo neprípustnej časti. Ak výsledkom prvého zobrazenia je kandidát porušujúci ohraničenia, tak tento kandidát je následne "opravený" pomocou druhého zobrazenia.

Výsledkom dekódovania nejakej reprezentácie je buď priamo kandidát, spĺňajúci všetky definované ohraničenia, alebo dvojica kandidátov, tvorená kandidátom z $\mathcal{M}_c^u$ (porušujúcim aspoň jedno z ohraničení) a jeho opravenou verziou z prípustnej časti $\mathcal{M}_c^f$. Počas evolučného vývoja teda dochádza ku generovaniu reprezentácií nevalidných kandidátov, ku ktorým je potrebné hľadať ich opravené verzie, spĺňajúce všetky definované ohraničenia.

Vhodne navrhnutá opravná procedúra g_2 by mala mať niekoľko vlastností:

- pre každého kandidáta z neprípustnej časti priestoru kandidátov by mala byť schopná nájsť jeho obraz v prípustnej časti,
- vzdialenosť medzi neprípustným kandidátom a jeho opravenou verziou by mala byť čo najmenšia,
- podobných neprípustných kandidátov by mala opravovať na podobné obrazy,
- transformácia by nemala byť výpočtovo náročná.

Prvá podmienka sa týka úplnosti – ak je možné opraviť ľubovoľného neprípustného kandidáta, potom algoritmus môže prehľadávať priestor prehľadávania bez akýchkoľvek obmedzení, je možné počas tohto prehľadávania vygenerovať ľubovoľnú reprezentáciu. Jej nedodržanie by kládlo dodatočné obmedzenia na pohyb v priestore prehľadávania alebo na konštrukciu dekodéra medzi priestorom prehľadávania a priestorom kandidátov.

Aj keď principiálne je možné nejakého neprípustného kandidáta opraviť na ľubovoľného kandidáta z prípustnej oblasti, druhá podmienka požaduje, aby medzi kandidátom a jeho opravenou verziou bola čo najväčšia podobnosť, aby opravená verzia sa od opravovaného kandidáta líšila čo najmenej a teda v čo najväčšej miere vyjadrovala tie vlastnosti, ktorých nositeľom je opravovaný kandidát.

Cieľ tretej vlastnosti, vyjadrujúcej princíp lokality, je rovnaký ako v prípade návrhu vhodného dekodéra v predchádzajúcej podkapitole.

Samotné využitie opravnej procedúry je jednoduché. Ak sa počas činnosti algoritmu vyskytne nejaký jedinec $a_i \in \mathcal{M}_c^u$, potom sa nájde jeho opravený obraz $a_i' \in \mathcal{M}_c^f$, kde $a_i' = g_2(a_i)$. Algoritmus následne pre neprípustného kandidáta používa vhodnosť $\Phi(a_i')$ s myšlienkou, že neprípustný kandidát je takým dobrým riešením, akým dobrým je to riešenie, na ktoré daný kandidát môže byť opravený. Neprípustný jedinec a_i v ďalšom pôsobí:

- naďalej ako a_i, iba sa preukazuje nie svojou vhodnosťou ale vhodnosťou svojej opravenej verzie a_i',
- je nahradený opravenou verziou a_i' (teda reprezentácia neprípustného jedinca je nahradená takou reprezentáciou, ktorá je dekódovaná pomocou g_1 priamo na opraveného kandidáta).

Pri náhrade kandidátov je otázne, aká časť zo všetkých neprípustných jedincov v populácii má byť skutočne nahradená svojimi opravenými verziami a aká časť má byť naďalej používaná vo svojej pôvodnej podobe[13].

Účinná opravná procedúra môže byť založená najmä na jednej z týchto troch[14] podôb:

- doménovo špecifická heuristika,

[13] Používanie jedincov s vhodnosťou, prislúchajúcou ich vylepšeným kópiám, sa označuje ako *Baldwinov efekt*, zatiaľ čo náhrada jedincov ich vylepšenými kópiami je označovaná ako *Lamarckova evolúcia*. Jediný rozdiel je v tom, že v týchto dvoch prístupoch pôvodne išlo o vylepšovanie jedincov, zatiaľ čo teraz ide o opravu neprípustných jedincov (opravený jedinec môže mať horšiu vhodnosť ako pôvodný neprípustný jedinec).

[14] Možné sú aj iné prístupy, napríklad [8] stavia na biologickej inšpirácii z oblasti samoopravných génových prenosov.

- náhodná oprava,
- systematická oprava.

V prípade doménovo špecifického prístupu nie je možné vytvoriť nejakú "všeobecnú" široko uplatniteľnú procedúru. Návrh takéhoto typu opravy je možné ľahko nájsť pre niektoré typy problémov, zatiaľ čo pre iné typy úloh sa jedná o zložitú úlohu – situácia je podobná ako pri návrhu vhodného dekodéra.

Na rozdiel od dekodéra, dekódovaného kandidáta je možné opravovať aj doménovo neutrálnym spôsobom – metódou pokus-omyl sú zložky kandidáta menené dovtedy, kým opravovaný kandidát porušuje aspoň jedno z definovaných ohraničení. Takáto oprava môže mať náhodný charakter (zložky aj ich nové hodnoty sú vyberané náhodným spôsobom) alebo zmeny je možné realizovať systematickým spôsobom (zložky a ich nové hodnoty sú skúmané v nejakom vopred určenom poradí). Vďaka tejto možnosti prístup založený na opravných procedúrach má širšie použitie ako prístup založený na dekodéroch, pretože môže byť použitý aj pre tie úlohy, kde využitie doménovo špecifických heuristík naráža na problémy.

Prístup založený na opravných procedúrach je pre oblasť kombinatorickej optimalizácie opäť možné ilustrovať na probléme obchodného cestujúceho. Uvažujme permutačnú reprezentačnú schému, pri ktorej je použitá taká reprezentácia jedinca, pri ktorej každý jedinec je reprezentovaný nejakou permutáciou hodnôt $\{v_1, \ldots, v_n\}$. Jedná sa teda o reprezentáciu vektorom s dĺžkou n, pričom na každej pozícii môže byť jedna z povolených hodnôt, avšak na rôznych pozíciách sa vyskytujú rôzne hodnoty. Príkladom takejto reprezentácie sú dva jedince

$$\begin{aligned} &v_2, v_3, v_8, v_7, v_9, v_4, v_1, v_5, v_6 \\ &v_2, v_3, v_8, v_7, v_9, v_6, v_4, v_5, v_1 \end{aligned} \tag{4.21}$$

Pre takúto reprezentáciu je možné použiť dekodér, ktorý výskyt hodnoty v_i chápe ako návštevu mesta i a súčasne rovnako aj pozíciu j chápe ako prítomnosť v meste j. Výskyt hodnoty v_i na pozícii j bude dekódovaný ako prechod z mesta j do mesta i [29]. Pri takomto dekódovaní by prvá reprezentácia bola dekódovaná na kandidáta tvaru

$$1 - 2 - 3 - 8 - 5 - 9 - 6 - 4 - 7 - 1 \tag{4.22}$$

kde cesta začína v meste 1, prechádza cez všetky mestá a záverečne sa vracia do mesta kde začala. Takýto kandidát spĺňa ohraničenia a preto patrí do prípustnej časti priestoru kandidátov – a teda nie je nutná jeho oprava.

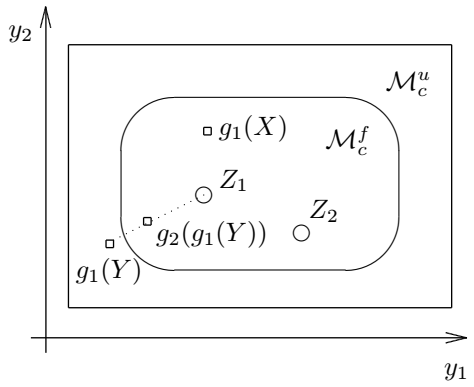
Použitie rovnakého spôsobu dekódovania pre druhú reprezentáciu vedie na kandidáta

$$1 - 2 - 3 - 8 - 5 - 9 - 1 \tag{4.23}$$

kde cesta sa predčasne uzavrie návratom do východzieho mesta bez toho, aby boli navštívené všetky mestá. Takýto kandidát je neprípustným a preto musí byť opravený.

Pre jeho opravu je možné použiť napríklad náhodný spôsob opravy – namiesto predčasného návratu do východzej pozície bude cesta rozšírená o návštevu jedného z tých miest, ktoré boli z cesty vynechané. Tento výber bude realizovaný náhodne[15]. Takýmto spôsobom je možné cestu opakovane rozširovať až po návštevu všetkých miest a získanie kandidáta, spĺňajúceho ohraničenia. Pri takomto spôsobe opravy pre náš príklad je možné namiesto návratu do mesta 1 pokračovať z mesta 9 do jedného z miest 4, 6 alebo 7. Ak by náhodne bolo vybrané mesto 6, tak ako ďalšie rozšírenie cesty je možné vybrať mesto 4 alebo 7, a po výbere jedného z nich je záverečne pridaný prechod do posledného doposiaľ nenavštíveného mesta. Takýmto spôsobom je náhodne zvolená jedna z $k!$ možných opráv (kde k reprezentuje počet miest, ktoré boli vynechané neprípustným kandidátom).

Opravné procedúry je možné využiť aj v oblasti numerickej optimalizácie [31]. Priestor kandidátov je v tomto prípade vytváraný ako kartézsky súčin domén reálne hodnotových premenných. Príklad založený na iba dvoch premenných je ilustrovaný na obr. 4.3.



Obr. 4.3: Oprava kandidátov porušujúcich ohraničenia.

[15]Pri snahe vyhnúť sa použitiu náhodného prvku v opravnej procedúre by bolo možné zvoliť prechod do toho z doposiaľ nenavštívených miest, do ktorého je najbližšie z mesta, ktoré momentálne je posledným mestom zaradeným do budovanej cesty.

Časť priestoru $\mathcal{M}_c$ je tvorená bodmi, ktoré predstavujú kandidátov, spĺňajúcich všetky ohraničenia. Ak reprezentácia je dekódovaná na bod z tejto časti, nie je nutná žiadna oprava. To je prípad reprezentácie X, ktorej dekódovaný obraz $g_1(X)$ leží vo vnútri prípustnej časti $\mathcal{M}_c^f$. Iná situácia je pri reprezentácii Y. Tento bod z priestoru prehľadávania je dekódovaný na kandidáta $g_1(Y)$, ktorý však leží v neprípustnej oblasti a musí byť preto opravovaný.

Princíp opravy je jednoduchý – stačí bodom $g_1(Y)$ preložiť priamku, ktorá je súčasne kolmá na hranicu prípustnej oblasti. Priesečník tej priamky s danou hranicou poskytuje bod, ktorý spĺňa všetky ohraničenia a súčasne je zo všetkých takýchto bodov najbližšie k pôvodnému $g_1(Y)$. Takže tento priesečník by bol vhodnou opravou. Problém je však v tom, že často hranica prípustnej oblasti nie je explicitne známa (resp. aj v prípadoch, keď známa je, môže byť žiadúce vyhnúť sa analytickým výpočtom).

Alternatívnym spôsobom k identifikácii prípustnej oblasti pomocou jej hranice je jej identifikácia pomocou bodov, ktoré do nej patria. Pre tento účel populácia jedincov (označovaná ako obyčajná populácia) je doplnená druhou populáciou, označovanou ako *referenčná* populácia.

Referenčná populácia pozostáva výlučne z jedincov, ktoré sú validné – spĺňajú všetky definované ohraničenia. Je inicializovaná náhodne, jedince sú náhodne generované. Ak generovaný jedinec porušuje aspoň jedno ohraničenie, tak sa zahodí a generovanie sa opakuje. Ak spĺňa všetky ohraničenia, tak je zaradený do referenčnej populácie. Takto sa nageneruje daný počet jedincov. Počet jedincov referenčnej populácie je predmetom voľby, z praktického hľadiska je zdola ohraničený hodnotou 1 a zhora veľkosťou obyčajnej populácie.

Ak je potrebné opraviť nejakého kandidáta $g_1(Y)$ majúceho tvar vektora súradníc $[y_1, y_2, \ldots, y_n]$, pretože porušuje niektoré z ohraničení, tak k tomuto jedincovi sa vyberie z referenčnej populácie referenčný jedinec Z_i majúci súradnice vyjadrené vektorom $[z_1, z_2, \ldots, z_n]$. Výber vhodného referenčného jedinca môže byť realizovaný ako:

- náhodný výber s rovnakými pravdepodobnosťami pre všetkých členov referenčnej populácie,
- výber toho referenčného jedinca, ktorého vzdialenosť k opravovanému kandidátovi je najmenšia,
- výber založený na vhodnosti, pri ktorom sú vhodnejšie jedince referenčnej populácie preferované voči menej vhodným.

Na spojnici bodov $g_1(Y)$ a Z_i sa určí bod $g_2(g_1(Y))$ so súradnicami $y'_1, y'_2, \ldots, y'_n$ použitím vzťahu

$$y'_i = cy_i + (1–c)z_i \tag{4.24}$$

pre určenie každej zložky bodu $g_2(g_1(Y))$. Hodnota c je z intervalu $< 0, 1 >$ (tá istá pre všetky zložky) a musí byť taká, aby výsledný bod (opravený kandidát) bol validný. Buď sa c generuje náhodne alebo sa v tejto úlohe deterministicky skúmajú vybrané hodnoty z jeho definičného intervalu v určenom poradí (napr. 1.0, 0.9, 0.8, ..., 0.0).

V ďalšom postupe sa buď používa pôvodná reprezentácia Y s vhodnosťou $\Phi(g_2(g_1(Y)))$ namiesto $\Phi(g_1(Y))$ alebo môže dôjsť k náhrade. V prípade náhrady sa reprezentácia Y nahradí za reprezentáciu Y' (ktorú je potrebné nájsť pre $g_2(g_1(Y))$ postupom inverzným k dekódovaniu), ktorá už bude vystupovať so svojou vlastnou vhodnosťou $\Phi(g_1(Y')) = \Phi(g_2(g_1(Y)))$. Aká veľká časť opráv kandidátov povedie na náhradu reprezentácií je otázkou nastavenia algoritmu pre riešenie konkrétnej úlohy[16].

V jednoduchšom prípade sa referenčná populácia nemení, ostáva tvorená stále tými istými jedincami. V zložitejšom prípade sa síce môže meniť, ale iba špeciálnym spôsobom – nie je nad ňou realizovaný evolučný proces. Jedným z možných spôsobov jej zmeny je prípad, keď vhodnosť $\Phi(g_2(g_1(Y)))$ je lepšia ako vhodnosť $\Phi(Z_i)$ – v tomto prípade referenčný jedinec Z_i môže byť nahradený jedincom $g_2(g_1(Y))$. Táto náhrada môže mať deterministický ale aj stochastický charakter.

4.1.4 Operátory zachovávajúce ohraničenia

Opäť je použitý evolučný algoritmus, ktorý môže mať štandardnú podobu, známu z riešenia úloh bez ohraničení. Dekódovanie je teraz jednokrokovým procesom, realizovaným ako zobrazenie

$$g : \mathcal{M}_s \rightarrow \mathcal{M}_c^f \cup \mathcal{M}_c^u \tag{4.25}$$

ktoré každý prvok z priestoru prehľadávania $\mathcal{M}_s$ zobrazuje na prvok z priestoru kandidátov, či už v jeho prípustnej alebo neprípustnej časti.

Principiálne, pri výbere príslušnej reprezentácie, je teda možné získať kandidáta, porušujúceho aspoň jedno z ohraničení. Aby sa vylúčila takáto možnosť, je algoritmus doplnený o mechanizmus udržiavania vhodného pohybu v priestore prehľadávania – algoritmus má povolené generovať iba také

[16] Autori prístupu doporučujú hodnotu 15% pre prípad numerickej optimalizácie.

reprezentácie, ktoré sú dekódované na kandidátov z prípustnej časti priestoru kandidátov $\mathcal{M}_c^f$. Takýmto spôsobom počas evolučného vývoja vôbec nedochádza k práci s neprípustnými jedincami, výsledkom dekódovania každej použitej reprezentácie je vždy priamo kandidát, spĺňajúci všetky definované ohraničenia.

Mechanizmus riadenia pohybu v priestore prehľadávania je reprezentovaný použitými operátormi, umožňujúcimi generovanie nových jedincov [30]. Tieto operátory musia zaručovať, že z rodičovských jedincov, ktorí sú dekódovaní na validných kandidátov, vzniknú nové jedince, ktoré budú taktiež dekódované do prípustnej časti priestoru kandidátov.

Je možné použiť štandardné operátory a v prípade vygenerovania reprezentácie, vedúcej na neprípustného kandidáta, túto reprezentáciu zahodiť a generovanie nového jedinca opakovať. Avšak efektívnejšie je použiť operátory bez nutnosti opakovaného generovania. Inak povedané, namiesto doménovo neutrálnych operátorov je potrebné použiť také operátory, ktoré rešpektujú ohraničenia, stanovené konkrétnym riešeným problémom.

Navyše, ohraničenia sú definované iba pre priestor kandidátov. Na základe týchto ohraničení je potrebné vytvoriť ohraničenia pre priestor prehľadávania, umožňujúce rozlíšiť reprezentácie, vedúce na validných kandidátov, od reprezentácií, dekódovaných na neprípustných kandidátov. Takáto transformácia ohraničení závisí od použitého spôsobu dekódovania – vo všeobecnosti, čím je dekódovanie jednoduchšie, tým je aj táto transformácia tiež jednoduchšia.

Prístup založený na špeciálnych operátoroch, zachovávajúcich platnosť ohraničení, má teda nasledujúce dve nevýhody:

- operátory sú doménovo špecifické, nie je možný návrh "všeobecných" operátorov, použiteľných pre viaceré navzájom rôzne úlohy,
- transformácia ohraničení z priestoru kandidátov do priestoru prehľadávania môže byť netriviálnou úlohou.

Na druhej strane je výhodou, že nedochádza ku generovaniu všetkých reprezentácií ale iba časti z nich. Keďže návrhom operátorov, zachovávajúcich ohraničenia, takýmto spôsobom dochádza vlastne k zmenšeniu priestoru prehľadávania, tak sa jedná o veľmi efektívny prístup.

Ako ilustračný príklad pre oblasť kombinatorickej optimalizácie je opäť možné použiť problém obchodného cestujúceho. Uvažujme mnohoznakovú reprezentačnú schému, pri ktorej je použitá taká reprezentácia jedinca, pri ktorej každý jedinec je reprezentovaný vektorom dĺžky n, pričom na každej pozícii sa môže vyskytovať jedna z hodnôt $\{v_1, \ldots, v_n\}$. V úlohe dekódovania

sa použije jednoduchý princíp – výskyt hodnoty v_i na pozícii i a hodnoty v_j na pozícii $i+1$ znamená, že z mesta i sa pokračuje do mesta j. Prvky v reprezentačnom vektore teda priamo vyjadrujú sekvenciu miest, cez ktoré cesta prechádza.

Vďaka triviálnosti dekódovania je možné ohraničenia, platné pre kandidátov, priamo preniesť na reprezentácie – dĺžka reprezentačného vektora musí byť stále n a ľubovoľné dve rôzne pozície reprezentačného vektora musia mať navzájom rôzne hodnoty.

Príkladom takejto reprezentácie sú dva jedince, pričom obaja sú dekódované na validných kandidátov

$$\begin{aligned} &v_8, v_3, v_2, v_1, v_4, v_9, v_5, v_7, v_6 \\ &v_2, v_8, v_7, v_4, v_9, v_3, v_1, v_6, v_5 \end{aligned} \tag{4.26}$$

Je zrejmé, že použitie mnohobodového alebo uniformného kríženia by s veľkou pravdepodobnosťou[17] viedlo k produkcii jedinca, predstavujúceho reprezentáciu, porušujúcu ohraničenie.

Jednou z možností je jedna z alternatív[18] OX (angl: order-based) kríženia [29]. Toto kríženie vyberá náhodne niekoľko pozícií v jednom rodičovi, pričom tieto pozície určujú poradie miest, ktoré sa na týchto pozíciách vyskytujú. A takéto isté poradie medzi danými mestami sa stanoví aj v druhom rodičovi, čím vznikne nový jedinec, ktorý bude mať prvky oboch rodičov a zároveň bude reprezentovať validnú cestu cez všetky mestá.

Predpokladajme, že náhodne boli zvolené pozície 3, 4, 6 a 8. Keďže druhý rodič na týchto pozíciách má hodnoty v_7, v_4, v_3 a v_6, tak z prvého rodiča sa preberú všetky ostatné hodnoty

$$v_8, ?, v_2, v_1, ?, v_9, v_5, ?, ? \tag{4.27}$$

a na prázdne miesta sa dopíšu vynechané hodnoty v takom poradí, v akom sú uvedené v druhom rodičovi, čím vznikne nový jedinec[19]

$$v_8, v_7, v_2, v_1, v_4, v_9, v_5, v_3, v_6 \tag{4.28}$$

Rovnakým postupom z druhého rodiča vzniká nový jedinec tvaru

$$v_2, v_8, v_1, v_4, v_9, v_3, v_7, v_6, v_5 \tag{4.29}$$

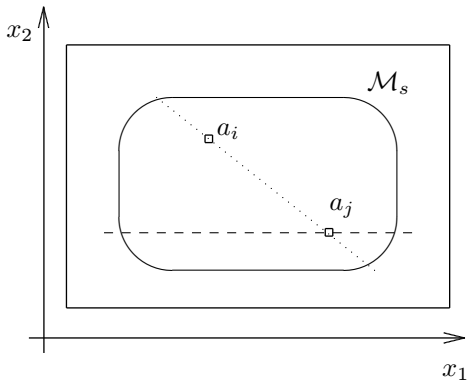
[17] V prípade jednobodového kríženia všetky možné výsledky sú neprípustné, neexistuje poloha deliaceho bodu vedúca na produkciu prijateľných potomkov.

[18] Verzia označovaná ako OX2.

[19] Pri výbere hodnôt, ktoré majú rovnaké vzájomné poradie v oboch jedincoch, nevzniká nový jedinec. Príkladom je výber pozícií 2, 4, 5 a 8, kde nový jedinec vzniká iba z druhého rodiča ale nie z prvého.

kde hodnoty v_2, v_1, v_9 a v_7 boli v štruktúre jedinca usporiadané v poradí, ktoré bolo dané prvým jedincom.

Použitie operátorov, zachovávajúcich ohraničenia, je možné ilustrovať aj pre úlohy numerickej optimalizácie [30]. Priestor prehľadávania je typicky tvorený kartézskym súčinom intervalov, prislúchajúcich jednotlivým pozíciám vektorovej reprezentácie jedincov. Príklad s dvomi premennými je na obr. 4.4. Predpokladá sa, že reprezentácie, dekódované na kandidátov spĺňajúcich všetky ohraničenia, vytvárajú jednu súvislú konvexnú oblasť[20] tak, ako to je naznačené.



Obr. 4.4: Použitie operátorov zachovávajúcich ohraničenia.

Typickou mutačnou zmenou v prípade bez existencie ohraničení je zmena hodnoty na jednej alebo niekoľkých pozíciách (v prípade zmien na viacerých pozíciách sú tieto zmeny navzájom nezávislé). Zmena na jednej pozícii znamená posun bodu, vyjadrujúceho reprezentáciu jedinca, v smere príslušnej súradnej osi.

Ak aj definičným oborom i-tej reprezentačnej pozície je nejaký interval $< u_i, v_i >$, neznamená to, že výsledkom mutačnej zmeny tejto pozície môže byť ľubovoľná hodnota z daného intervalu. Prípustnou zmenou je hodnota z menšieho intervalu, zaručujúceho, že jedinec ostáva naďalej v žiadanej oblasti. Hranice tohto intervalu sú dané ako priesečníky hranice oblasti a priamky, prechádzajúcej daným bodom a zároveň rovnobežnej s i-tou súradnou osou.

Problémom je to, že hranice zmenšeného intervalu závisia od hodnôt na ostatných reprezentačných pozíciách a je ich teda nutné určovať osobitne

[20] V prípade, že všetky ohraničenia majú tvar lineárnych nerovníc, tak oblasť prípustných reprezentácií je spojitá a konvexná.

pre každú pozíciu každého jedinca. Je ich možné určovať analyticky alebo pomocou vzťahu

$$cu_i + (1 - c)v_i \tag{4.30}$$

kde sa hľadajú také dve hodnoty parametra c z intervalu $< 0, 1 >$, ktorým zodpovedajú prechody cez hranicu prípustnej oblasti. Hľadanie môže byť náhodné alebo systematické (zmena hodnoty o nejaký inkrement).

Po určení hraníc zmenšeného intervalu je možné použiť štandardnú mutáciu s uvažovaním nájdených hraníc (uniformnú, extremálnu, neuniformnú, atď. [26]). Je však potrebné vziať do úvahy, že zmenou hodnoty na nejakej pozícii sa súčasne môžu meniť hranice prípustných intervalov pre iné pozície toho istého jedinca.

Často používanou formou krížiaceho operátora je aritmetické kríženie, ktoré môže mať niekoľko rôznych podôb [26]. Ak sa využije vlastnosť konvexnej oblasti, že každý bod na spojnici ľubovoľných dvoch bodov z tejto oblasti patrí tiež do danej oblasti, tak je možné použiť aritmetické kríženie v tvare, keď potomok sa určuje podľa vzťahu (pre i-tu pozíciu)

$$x_i = \chi x_i^{a_1} + (1 - \chi)x_i^{a_2} \tag{4.31}$$

kde χ je náhodné číslo z intervalu $< 0, 1 >$, pričom rovnaká hodnota χ sa použije pre všetky reprezentačné pozície nového jedinca. Tento typ kríženia dokáže produkovať potomkov na úsečke, ohraničenej bodmi, reprezentujúcimi oboch rodičov. Je možné produkovať aj nových jedincov mimo spojnice oboch rodičov (napr. doplnením násobenia zložkami perturbačného vektora), avšak v tomto prípade je potrebné ešte doplniť kontrolu, či vygenerovaný jedinec skutočne leží v prípustnej oblasti.

V prípade, že prípustná oblasť nie je konvexná alebo nie je spojitá, môže sa stať, že nie každá hodnota zo zmenšeného intervalu hodnôt, prislúchajúceho nejakej reprezentačnej pozícii, alebo nie každý bod na spojnici, spájajúcej dvoch jedincov, ležia v prípustnej oblasti. Najjednoduchším postupom v tomto prípade je následná kontrola prípustnosti a v prípade porušenia niektorého z ohraničení opakované generovanie. O efektívnosti použitia operátorov, zachovávajúcich ohraničenia, v tomto prípade rozhoduje pomer odmietnutých a prijatých nových jedincov.

4.2 Hľadanie na hranici prípustnej oblasti

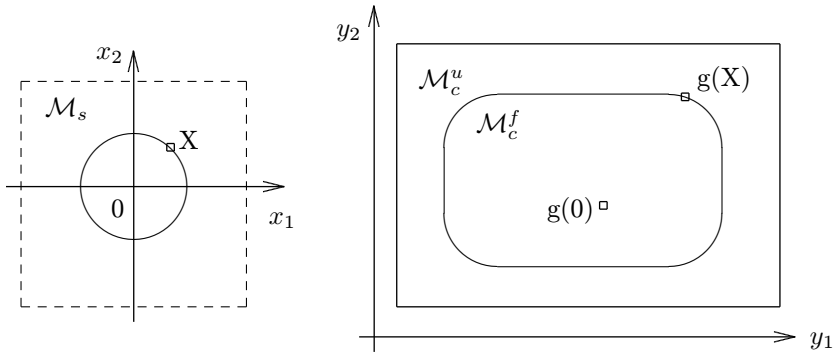
Prekvapivo často sa vyskytuje situácia, keď hľadané riešenie leží presne na hranici prípustnej časti priestoru kandidátov $\mathcal{M}_c^f$. Príkladom je numerická

optimalizácia, keď funkcia vhodnosti má tvar lineárnej funkcie. Iným prípadom môže byť vhodnosť nelineárneho unimodálneho tvaru s výskytom extrému za hranicou povolenej oblasti.

Aj keď je možné priamo použiť uvedené metódy riešenia úloh s ohraničeniami, tieto sa musia vysporadúvať s faktom, že v okolí hľadaného riešenia je veľká časť kandidátov neprípustných – kandidátov patriacich do $\mathcal{M}_c^u$. Napríklad pri použití penalizačných funkcií pri slabšej penalizácii algoritmus môže skonvergovať v neprípustnej časti ešte pred dosiahnutím hranice, zatiaľ čo použitie silnejšej penalizácie môže zapríčiniť presunutie konvergencie do vnútra prípustnej oblasti.

Preto sú pochopiteľné snahy o explicitné zohľadnenie predpokladanej polohy hľadaného riešenia. Pri použití penalizácie to je rozlišovanie medzi externou a internou penalizáciou [42]. Externá penalizácia je zapríčinená porušovaním všetkých alebo iba niektorých z definovaných ohraničení (teda polohou kandidáta v neprípustnej časti priestoru kandidátov). Interná penalizácia je naopak vyvolaná polohou kandidáta vo vnútri prípustnej oblasti $\mathcal{M}_c^f$ – a je tým väčšia, čím je kandidát hlbšie v tejto oblasti.

Zaujímavým prístupom, kombinujúcim operátory, zachovávajúce ohraničenia, s metódou dekodérov je prístup uvedený v [40]. Celý prístup je znázornený na obr. 4.5.



Obr. 4.5: Hľadanie riešenia na hranici prípustnej oblasti.

Prehľadávanie priestoru $\mathcal{M}_s$ je riadené tak, aby algoritmus pracoval iba s tými reprezentáciami, ktoré sú dekódované na kandidátov, ktorí spĺňajú všetky ohraničenia a zároveň ležia na hranici medzi prípustnou a neprípustnou oblasťou priestoru kandidátov.

Je výhodné, ak takéto reprezentácie v priestore prehľadávania vytvárajú pravidelný útvar, pretože to zjednodušuje návrh genetických operátorov.

Jedným z takýchto tvarov je sféra s polomerom r. Ak tento polomer bude $r = 1$[21] a súčasne priestor prehľadávania je n-rozmerný, potom táto sféra je tvorená všetkými bodmi $X = [v_1, v_2, \ldots, v_n]$ priestoru $\mathcal{M}_s$, pre ktoré platí

$$\sum_{i=1}^{n} (v_i)^2 = 1 \tag{4.32}$$

Na začiatku algoritmus potrebuje inicializovať svoju populáciu reprezentáciami, ktoré ležia na tejto sfére. Je to možné realizovať jednoduchým náhodným generovaním a následnou opravou jedincov tak, aby spĺňali uvedenú podmienku. Ak bol v procese inicializácie vygenerovaný jedinec v tvare vektora hodnôt $[v_1, v_2, \ldots, v_n]$, potom takéhoto jedinca je možné opraviť na tvar

$$\left[\frac{v_1}{\sqrt{\sum_{i=1}^{n} (v_i)^2}}, \ldots, \frac{v_n}{\sqrt{\sum_{i=1}^{n} (v_i)^2}} \right] \tag{4.33}$$

Pre generovanie jedincov je potrebné použiť operátory, ktoré z rodičov, ležiacich na jednotkovej sfére, vytvoria potomkov, ktoré tiež ležia na danej sfére. V úlohe rekombinačného operátora je možné použiť *sférické* kríženie, pri ktorom z dvoch rodičov $[u_1, u_2, \ldots, u_n]$ a $[v_1, v_2, \ldots, v_n]$ vzniká nový jedinec tvaru

$$\left[\sqrt{\alpha(u_1)^2 + (1-\alpha)(v_1)^2}, \ldots, \sqrt{\alpha(u_n)^2 + (1-\alpha)(v_n)^2} \right] \tag{4.34}$$

kde hodnota α je náhodne vybratá z intervalu $< 0, 1 >$.

V úlohe mutačného operátora je možné použiť *sférickú* mutáciu, ktorá transformuje rodičovského jedinca $[v_1, v_2, \ldots, v_n]$ na nového jedinca výberom dvoch pozícií i a j a následnou úpravou týchto pozícií podľa

$$v_i \rightarrow \alpha v_i \qquad v_j \rightarrow v_j \sqrt{\left(\frac{v_i}{v_j} \right)^2 (1 - \alpha^2) + 1} \tag{4.35}$$

pričom hodnota α je opäť náhodne vybratá z intervalu $< 0, 1 >$.

Dekódovanie možno realizovať podobným spôsobom ako pri metóde dekodérov. Bod X sa premietne na okraj hyperkocky, v ktorej je kružnica vpísaná, a následne sa tento priemet dekóduje na kandidáta, nachádzajúceho sa na okraji priestoru kandidátov. Na spojnici dvoch kandidátov (kandidát na okraji priestoru a kandidát $g(0)$, na ktorý je dekódovaný stred priestoru prehľadávania) sa nájde ten, ktorý leží na hranici prípustnej oblasti, a tento kandidát bude zodpovedať skúmanej reprezentácii X.

[21] Pre zjednodušenie v ďalšom bude uvažovaný iba tento špeciálny prípad bez toho, aby to bolo opakovane zdôrazňované.

4.3 Porovnanie metód pre riešenie úloh s ohraničeniami

Pre empirické porovnanie niekoľkých vybraných metód, schopných zohľadniť prítomnosť ohraničení, bola použitá minimalizačná úloha z oblasti numerickej optimalizácie. Kvôli možnosti názornej vizualizácie sa jednalo o úlohu iba v dvojrozmernom priestore – ako testovacia funkcia bola použitá nelineárna funkcia dvoch premenných

$$f(x_1, x_2) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{4.36}$$

pričom obe premenné boli uvažované z intervalu $< 0, 25 >$. Priebeh funkcie je zobrazený na obr. 4.6 (hore). V uvažovanej časti priestoru má funkcia monotónny tvar s minimom v bode $[0, 0]$.

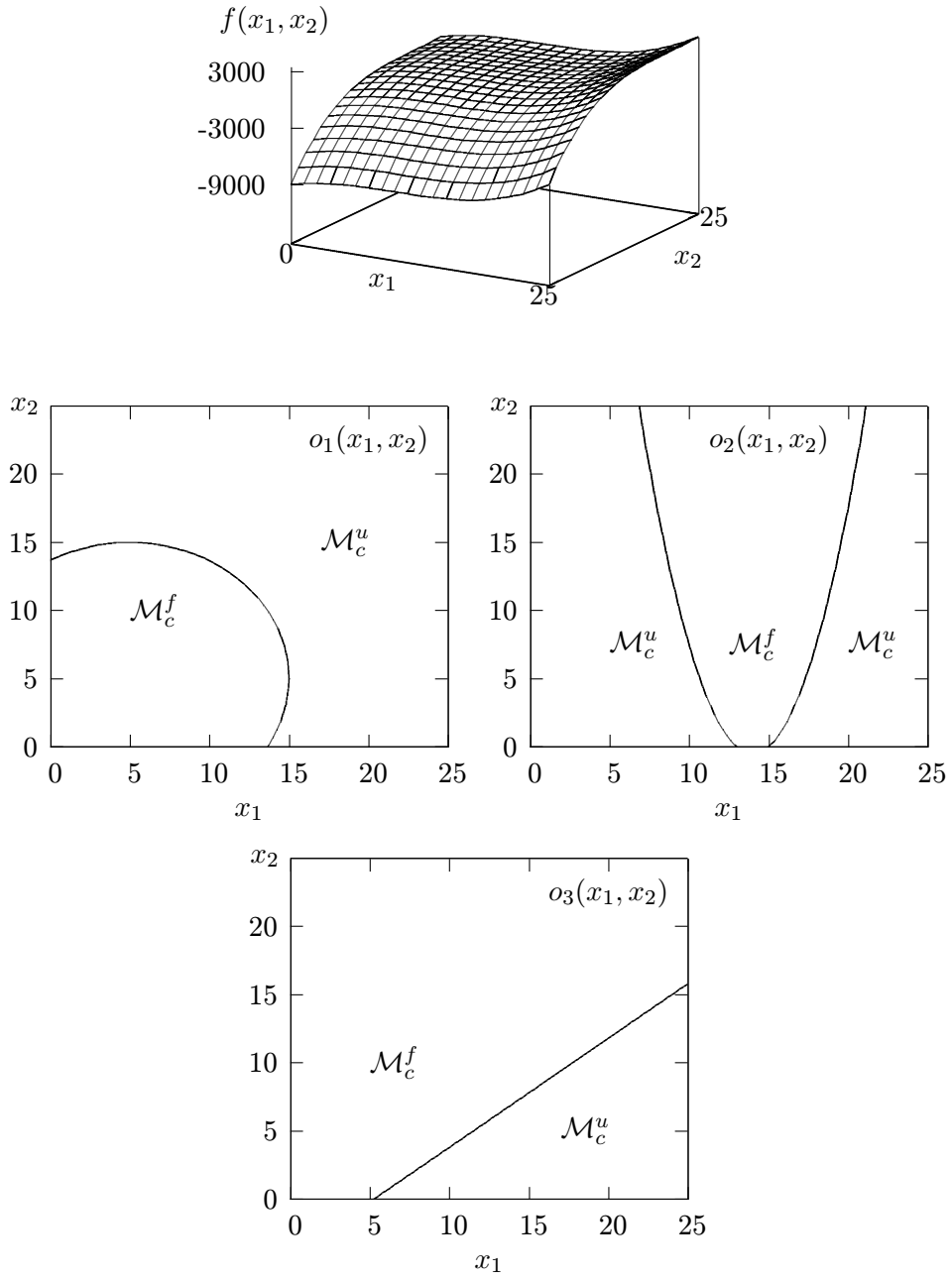
Úlohou bolo nájsť minimum tejto funkcie v prípustnej oblasti, ktorá bola daná pomocou sady troch ohraničení v tvare nerovností

$$\begin{array}{lcrcl} o_1(x_1, x_2) & : & (x_1 - 5)^2 + (x_2 - 5)^2 - 100 & \leq & 0 \\ o_2(x_1, x_2) & : & 0.5(x_1 - 14)^2 - x_2 - 0.5 & \leq & 0 \\ o_3(x_1, x_2) & : & 0.8x_1 - x_2 - 4 & \leq & 0 \end{array} \tag{4.37}$$

Každé z týchto ohraničení delí priestor na prípustnú $\mathcal{M}_c^f$ a neprípustnú oblasť $\mathcal{M}_c^u$. Tieto delenia sú taktiež znázornené na obr. 4.6 pre každé z ohraničení osobitne. Výsledná prípustná oblasť je prienikom prípustných oblastí jednotlivých ohraničení. Je umiestnená tak, že minimum testovacej funkcie leží mimo tejto oblasti.

Vzhľadom na polohu minima funkcie f a rýchlosti klesania je možné jednotlivé ohraničenia usporiadať podľa obtiažnosti ich splnenia. Najjednoduchšie splniteľným je ohraničenie o_1 (obr. 4.6 stred vľavo). Konvergencia algoritmu na základe tvaru funkcie f má za následok presúvanie jedincov z neprípustnej oblasti do oblasti prípustnej bez nutnosti toto ohraničenie špeciálne zohľadňovať. Ohraničenie takéhoto typu nerobí algoritmu žiadny problém.

O niečo obtiažnejšie je ohraničenie o_3. V konečnom dôsledku konvergencia k minimu v bode $[0, 0]$ taktiež vedie k splneniu tohto ohraničenia, avšak situácia je trochu odlišná počas procesu evolučného hľadania. Príčinou je to, že vo veľkej časti priestoru funkcia f klesá oveľa rýchlejšie v smere poklesu hodnôt x_2 než v smere poklesu hodnôt x_1. Následkom toho napríklad jedinec s hodnotami premenných $[10, 5]$ má väčšiu tendenciu k pohybu dole než doľava a s veľkou pravdepodobnosťou sa presunie z prípustnej časti priestoru do neprípustnej časti (obr. 4.6 dole). Takýmto spôsobom počas evolučného



Obr. 4.6: Testovacia funkcia a delenie priestoru na prípustnú a neprípustnú časť definovanými ohraničeniami.

hľadania môže dochádzať k zvyšovaniu počtu tých jedincov, ktoré porušujú ohraničenie o_3.

Najobtiažnejším je ohraničenie o_2 (obr. 4.6 stred vpravo). Je zrejmé, že tendencia algoritmu presúvať jedince v populácii k bodu $[0, 0]$ zapríčiňuje nesplnenie tohto ohraničenia. Bez modifikácie algoritmu pre zohľadnenie takéhoto typu ohraničenia by algoritmus vždy konvergoval do neprípustnej časti priestoru kandidátov.

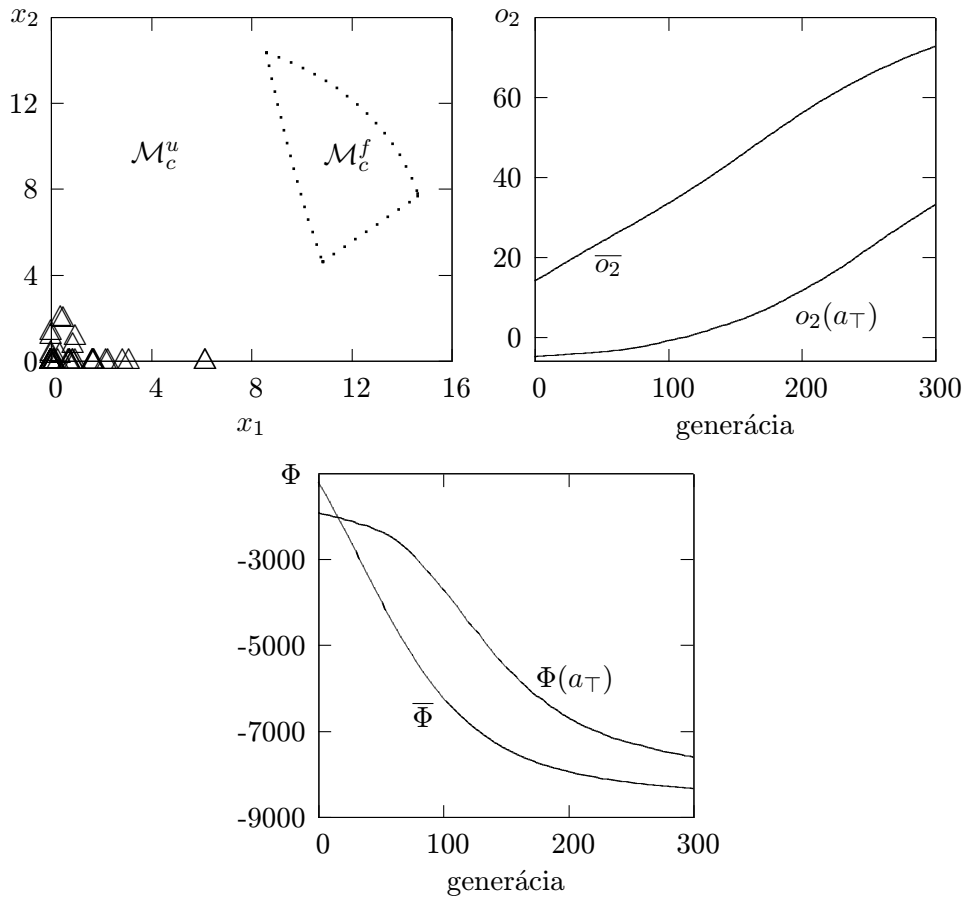
Ako testovacie prostredie bol použitý jednoduchý tvar testovacieho algoritmu, ktorý využíval nasledujúce prvky:

- reprezentáciu pomocou vektora reálnych čísel (celkovo boli použité 2 premenné),
- náhodnú inicializáciu populácie (o veľkosti 50) jedincov,
- selekciu 2 rodičov binárnym turnajom bez náhrady,
- tvorbu potomkov jednobodovým krížením ($p_c = 0.5$) a normálnou mutáciou pre zmenu hodnoty ($p_m = 1/2$),
- tvorbu novej generácie náhradou náhodne vybraných jedincov populácie novovytvorenými jedincami.

Štruktúra algoritmu umožňovala tento základný tvar rozširovať o rôzne metódy pre zohľadnenie definovaných ohraničení (boli zohľadňované všetky tri ohraničenia).

Chovanie testovacieho algoritmu (uvažujúceho iba funkciu f bez zreteľa na ohraničenia) je znázornené na obr. 4.7. Sledovalo sa rozloženie jedincov v priestore kandidátov v poslednej generácii (vľavo hore), priebeh plnenia najťažšieho ohraničenia o_2 (vpravo hore) a priebeh hodnôt vhodnosti, priamo reflektujúcej hodnoty testovacej funkcie (dole). Pri priebehoch sú indikované dve hodnoty – priemerná hodnota v populácii a hodnota najlepšieho[22] jedinca populácie. Priebehy vyjadrujú priemerné hodnoty – bolo vykonaných 1000 opakovaní a dosiahnuté výsledky boli spriemernené. Naproti tomu priestorová distribúcia jedincov v poslednej generácii je výsledkom iba jedného opakovania evolučného hľadania a je potrebné ju chápať ako ukážku typického rozloženia (pri rôznych realizáciách hľadania sú získavané rozloženia podobné, avšak líšiace sa v presnom umiestnení jedincov). Pre lepšiu orientáciu je hranica prípustnej časti priestoru vyznačená bodkovane.

[22] Jedinec, ktorý spĺňa všetky ohraničenia, je považovaný za lepšieho ako ten, ktorý porušuje niektoré ohraničenia. Ak z dvoch jedincov sú oba prípustné, lepším je ten, ktorý má menšiu vhodnosť. Ak dva jedince sú neprípustné, lepším je ten, ktorý menej porušuje ohraničenia.



Obr. 4.7: Výsledky pre testovací algoritmus.

Otázkou je, ako by mali vyzerať uvedené zobrazenia v prípade použitia "dobrej" metódy pre riešenie úloh s ohraničeniami. Zmeny by mali byť:

- jedince (alebo aspoň časť z nich) by mali byť vo vnútri prípustnej oblasti a nie ďaleko za jej hranicou (populácia by mala konvergovať do prípustnej oblasti),
- hodnota ohraničenia najlepšieho jedinca v populácii (označovaného $a_\top$) by mala byť menšia nanajvýš rovná 0 (inak v populácii neexistuje jedinec, ktorý by spĺňal všetky ohraničenia),
- vhodnosť najlepšieho jedinca by mala byť čo najmenšia (blížiaca sa hodnote -3626).

To, či priemerná hodnota ohraničenia je umiestnená pod alebo nad hodnotou 0 závisí od toho, koľko jedincov je v prípustnej oblasti a koľko mimo nej (a ako hlboko sú tieto jedince v neprípustnej oblasti).

Priemerná vhodnosť v populácii môže byť pod alebo nad vhodnosťou najlepšieho jedinca. Umiestnenie pod ňou signalizuje prítomnosť jedincov ležiacich mimo prípustnú oblasť (čím je $\overline{\Phi}$ hlbšie pod $\Phi(a_\top)$, tým je týchto jedincov viac alebo sú vzdialenejšie od hranice prípustnej oblasti).

Do použitého testovacieho algoritmu boli pridávané tieto metódy pre riešenie úloh s ohraničeniami:

Sekvenčné spĺňanie ohraničení. Hľadanie riešenia prechádza viacerými fázami (ich počet je o jednu väčší ako počet definovaných ohraničení) [41]. V prvej fáze v úlohe vhodnosti vystupuje miera porušenia prvého ohraničenia a trvá až dovtedy, kým počet jedincov v populácii, ktoré spĺňajú prvé ohraničenie, presiahne zadaný prah. Vyprodukovaná populáciu jedincov je vstupom do ďalšej fázy.

V každej ďalšej fáze (okrem poslednej) v úlohe vhodnosti bude vystupovať miera porušenia jedného z ohraničení. Za rodičov môžu byť volené iba tie jedince, ktoré spĺňajú všetky predchádzajúce ohraničenia. Hľadanie začína populáciou, ktorá bola výsledkom predchádzajúcej fázy, a končí vtedy, keď počet jedincov spĺňajúcich práve spracovávané ohraničenie presiahne zadaný prah. Pre výsledky zobrazené na obr. 4.8 vľavo bol použitý prah 75% veľkosti populácie. Výstupná populácia je opäť vstupom do ďalšej fázy.

V záverečnej fáze sa použije funkcia vhodnosti v pôvodnej podobe a za rodiča môže byť vybraný iba jedinec, spĺňajúci všetky ohraničenia.

Pri vkladaní novo vytvoreného jedinca do populácie pri formovaní novej generácie vkladaný jedinec nahradí prednostne takého jedinca, ktorý už nemôže byť v danej fáze vybratý za rodiča (porušuje aspoň jedno z predchádzajúcich ohraničení).

Viackriteriálna optimalizácia. Problém bol transformovaný do tvaru úlohy pre dvojkriteriálnu optimalizáciu. Prvým kritériom bola pôvodná vhodnosť, druhé kritérium reprezentoval nevážený súčet porušení jednotlivých ohraničení (toto kritérium vyžadovalo minimalizáciu).

Ako riešiaca metóda bola použitá metóda NSGA-II (a fast elitist Nondominated Sorting Genetic Algorithm – jej popis je uvedený v kap. 2).

Statická penalizácia. Metóda penalizačných funkcií s úpravou vhodnosti na tvar podľa vzťahu (4.9). Pre výsledky zobrazené na obr. 4.9 vľavo bolo

použité nelineárne zohľadňovanie veľkosti porušení jednotlivých ohraničení (s nastavením $\alpha = 2$).

Rôzna obtiažnosť splnenia jednotlivých ohraničení bola vyjadrená rôznymi váhami. Tieto boli na základe početných pokusných experimentov s dosahovaním riešenia pre rôzne kombinácie hodnôt váhových koeficientov nastavené na hodnoty $w_1 = 500$, $w_2 = 2000$ a $w_3 = 1000$.

Dynamická penalizácia. Metóda penalizačných funkcií s dynamicky meneným pomerom medzi vhodnosťou a porušením jednotlivých ohraničení, pričom s časom dôležitosť spĺňania ohraničení sa zvyšuje. Vhodnosť bola upravovaná podľa vzťahu

$$\Phi(a_i) + \left(\frac{t}{2}\right)^{\alpha} \sum_{j=1}^{n} (\Delta_j(a_i))^{\alpha} \tag{4.38}$$

kde t reprezentuje čas (označuje aktuálnu generáciu). Výsledky na obr. 4.9 vpravo boli získané pri nastavenej nelineárnej závislosti ($\alpha = 2$).

Dekodér. Princíp dekódovania pre numerickú optimalizáciu v podobe ako bol ilustrovaný na obr. 4.2 a popísaný v sprievodnom texte. Úlohu obrazu $g(0)$ hral referenčný kandidát spĺňajúci všetky ohraničenia, náhodne generovaný pred samotným hľadaním riešenia.

Opravná procedúra. Princíp opravnej procedúry pre numerickú optimalizáciu v podobe ako bol ilustrovaný na obr. 4.3 a popísaný v sprievodnom texte.

Výsledky zobrazené na obr. 4.10 vpravo boli získané pri referenčnej populácii pozostávajúcej z desiatich náhodne generovaných prípustných jedincov, pričom jej zloženie sa nemenilo. Výber vhodného referenčného jedinca bol realizovaný na základe vzdialenosti k opravovanému kandidátovi. Oprava kandidáta mala za následok aj náhradu reprezentácie tohto kandidáta s pravdepodobnosťou 0.5.

Operátory zachovávajúce ohraničenia. Populácia je náhodne inicializovaná takým spôsobom, aby každý jedinec v populácii spĺňal všetky definované ohraničenia (náhodné generovanie sa v prípade nutnosti opakuje).

Počas evolučného hľadania sa používa štandardná podoba aritmetického kríženia. V úlohe mutácie bola použitá ako neuniformná tak aj extremálna mutácia, pričom obe zohľadňovali aktuálnu podobu intervalu povolených

hodnôt. Výsledky na obr. 4.11 vľavo boli získané pri výbere mutačného operátora z dostupných alternatív s rovnakou pravdepodobnosťou.

Hľadanie na hranici prípustnej oblasti. Pracuje sa iba s takými reprezentáciami v priestore prehľadávania, ktoré ležia na obvode jednotkovej kružnice. Pri inicializácii populácie sú náhodne vygenerované jedince normalizované podľa vzťahu (4.33) a nové jedince sú vytvárané použitím sférickej mutácie definovanej vzťahom (4.35). Krížiaci operátor nebol použitý. Vytvárané reprezentácie sú dekódované na kandidátov, ležiacich na hranici prípustnej oblasti.

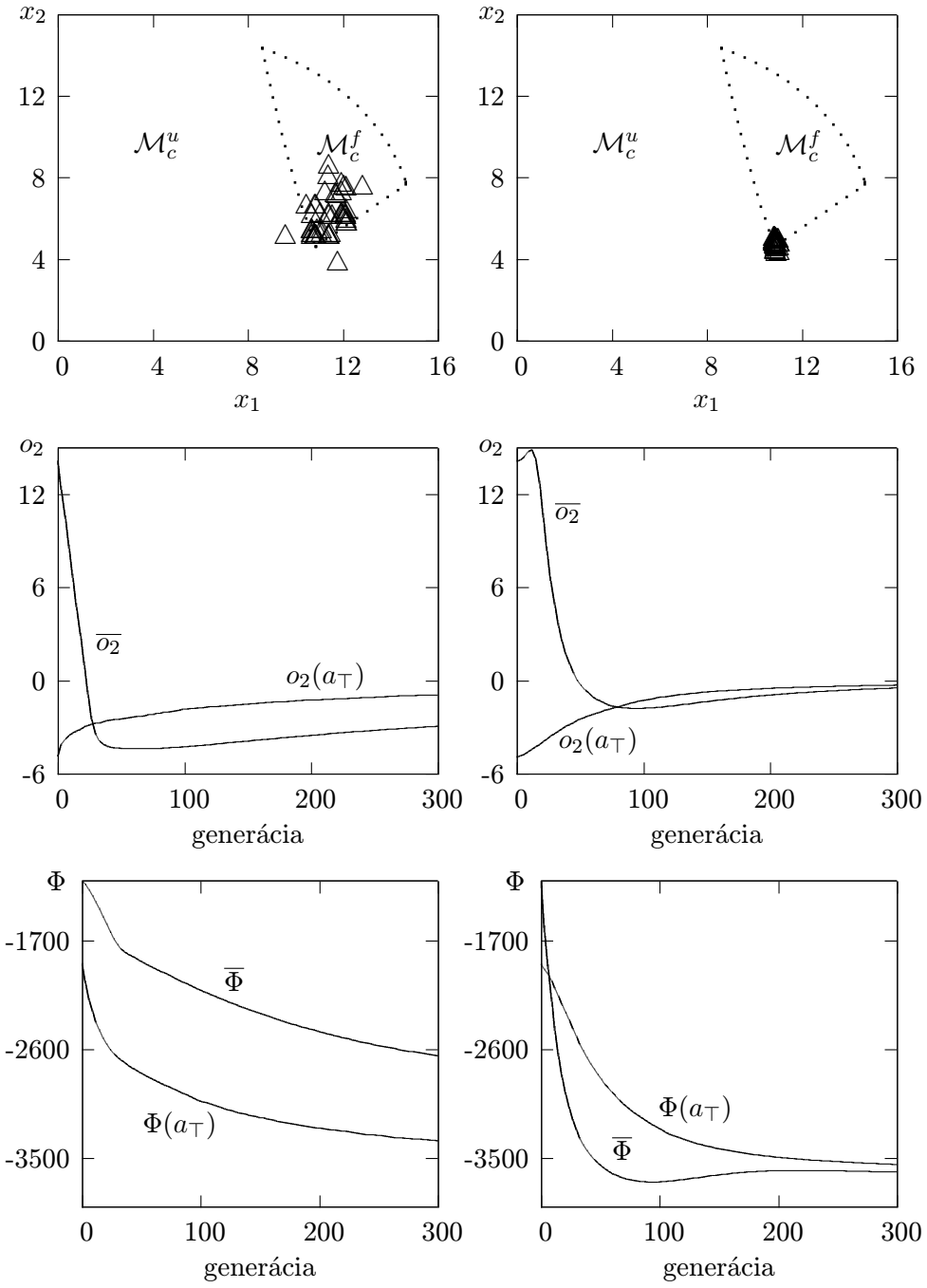
Výsledky dosiahnuté uvedenými testovanými metódami sú zobrazené na obr. 4.8 až obr. 4.11. Metódy viackriteriálnej optimalizácie, dekodérov a hľadania na hranici prípustnej oblasti nevyžadovali nastavenie riadiacich parametrov, ostatné metódy požadovali nastaviť aspoň jednu hodnotu. Toto nastavenie bolo pokusne realizované tak, aby dané metódy vykazovali pre testovaciu funkciu čo najlepšie chovanie.

Z hľadiska konvergencie populácie sa výrazne od iných metód vyčleňujú viackriteriálna optimalizácia a použitie špecializovaného dekodéru. V oboch prípadoch populácia skonvergovala do úzkeho podpriestoru v okolí hľadaného riešenia. O niečo menší stupeň konvergencie vykazuje použitie operátorov zachovávajúcich ohraničenia, pri ktorej sa populácia rozpadla na dve časti – horná skupina jedincov vďačí za svoju existenciu použitiu extremálnej mutácie.

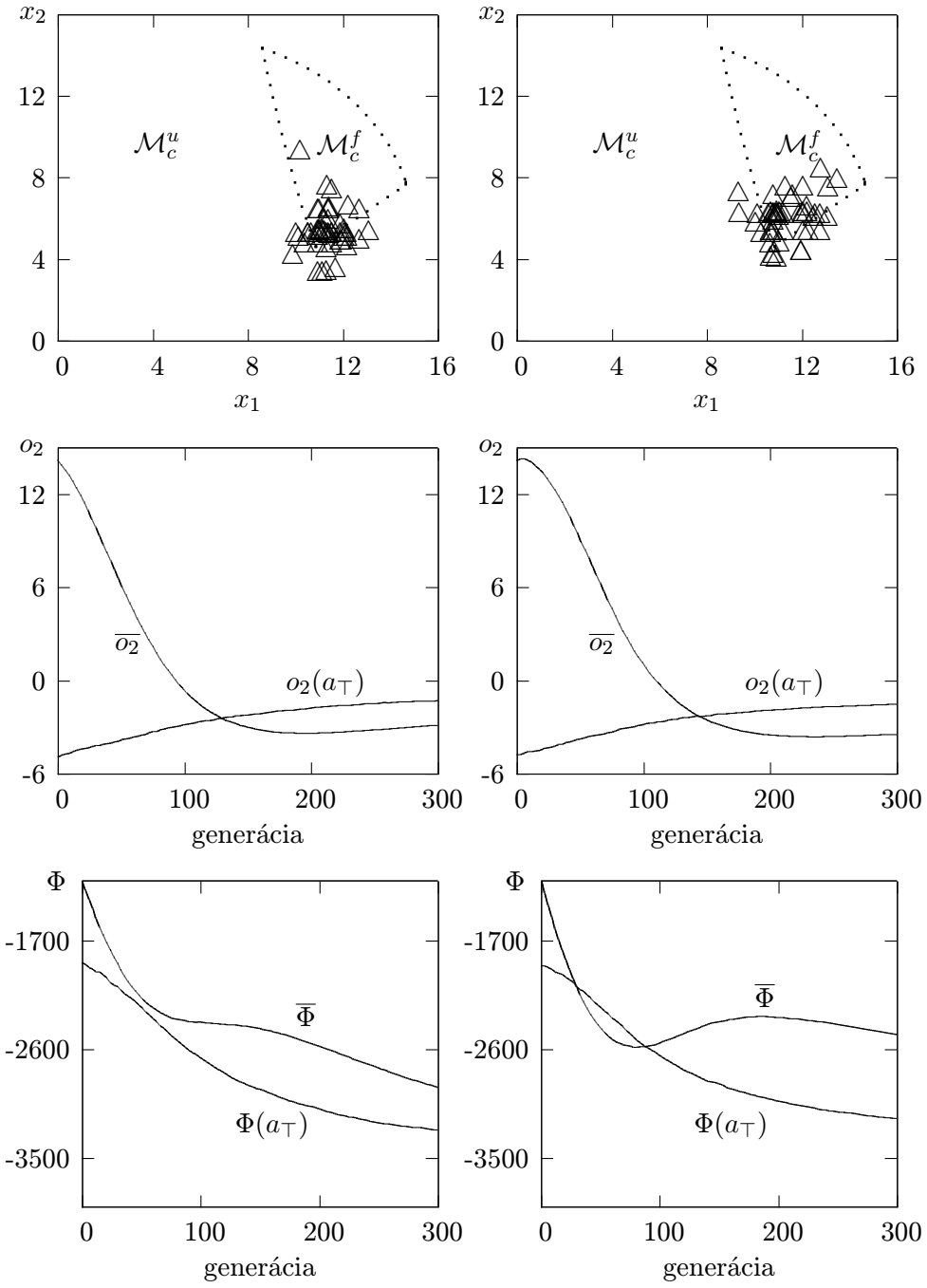
Ostatné metódy konvergujú v menšej miere, jedince sú rozptýlené na väčšej ploche. Pritom niekedy je mimo prípustnej oblasti menej jedincov (metóda sekvenčného spĺňania) a inokedy zase jedincov viac (penalizačné metódy). Hľadanie na hranici prípustnej oblasti prirodzene rozmiestňuje jedince iba pozdĺž línií, reprezentujúcich danú hranicu.

Skúmanie priebehu $o_2(a_\top)$ odhalí podobnosť všetkých metód – priebeh začína na zápornej hodnote a s pribúdajúcimi generáciami sa blíži hodnote 0. Teda najlepší jedinec sa nachádza vo vnútri prípustnej oblasti a postupne sa približuje k jej hranici. Rozdiel medzi metódami je v rýchlosti približovania sa k hranici a v dosiahnutej vzdialenosti (hľadané riešenie leží na hranici prípustnej oblasti). Z tohto hľadiska najhoršie výsledky dosiahli penalizačné metódy spolu so sekvenčným spĺňaním ohraničení.

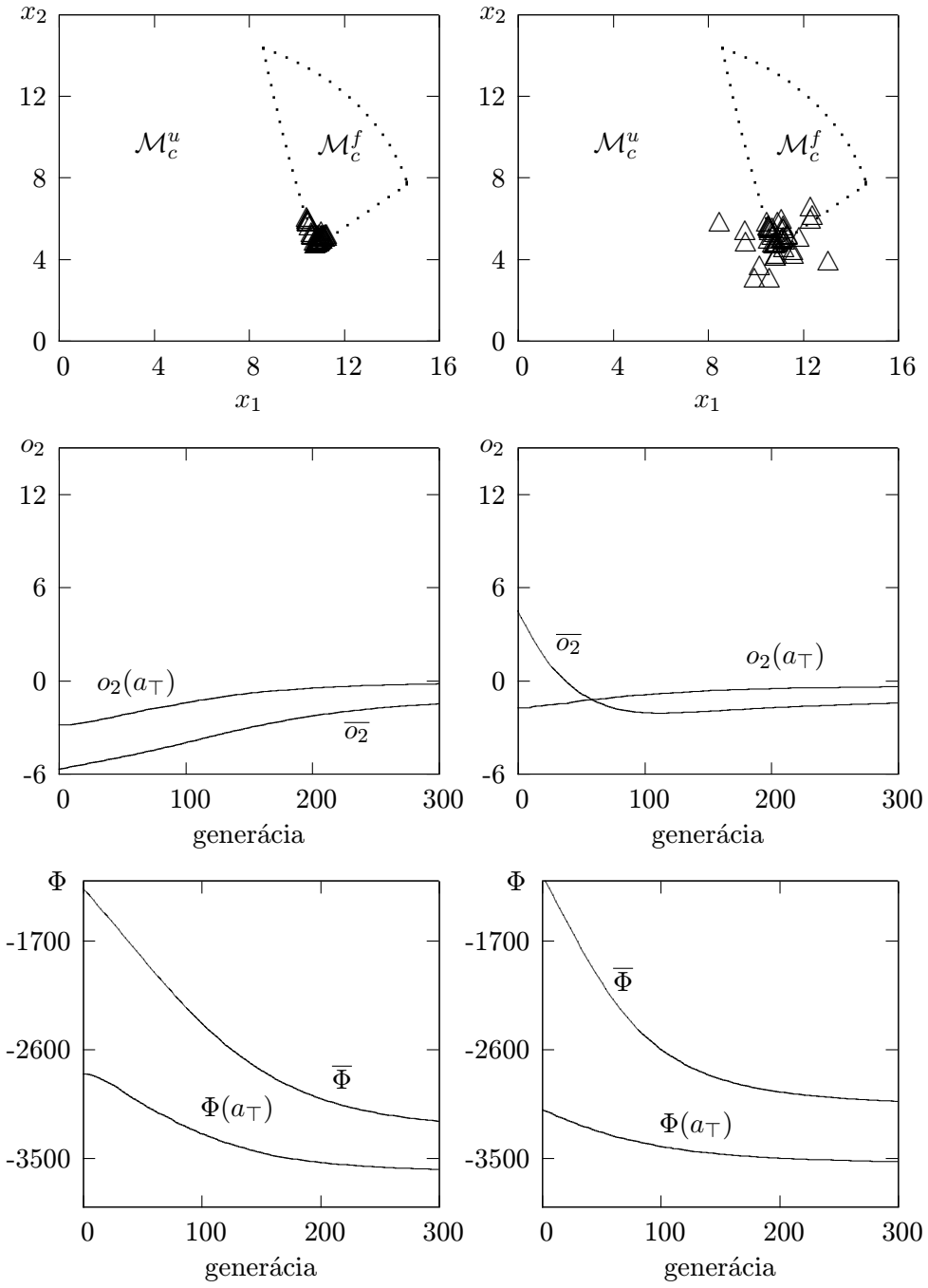
Keďže metódy špecializovaného dekodéru a operátorov zachovávajúcich ohraničenia generujú kandidátov iba v prípustnej oblasti, tak aj ich priebeh $\overline{o_2}$ leží v záporných hodnotách počas celého hľadania. Podobne to je aj



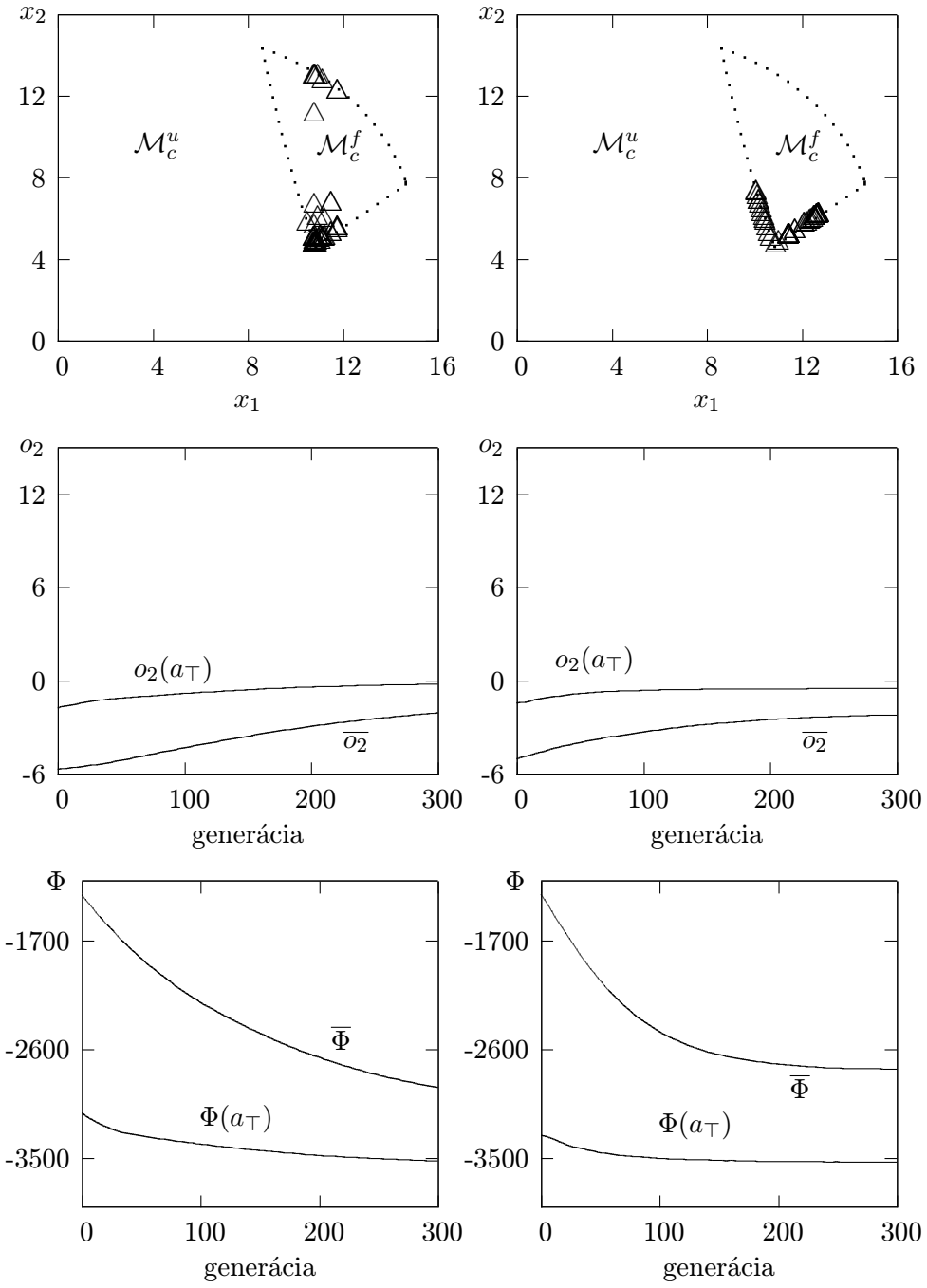
Obr. 4.8: Výsledky pre metódu sekvenčného spĺňania ohraničení (vľavo) a viackriteriálnej optimalizácie (vpravo).



Obr. 4.9: Výsledky pre metódu statickej penalizácie (vľavo) a dynamickej penalizácie (vpravo).



Obr. 4.10: Výsledky pre metódu špeciálneho dekodéra (vľavo) a opravnej procedúry (vpravo).



Obr. 4.11: Výsledky pre metódu operátorov zachovávajúcich ohraničenia (vľavo) a hľadania na hranici prípustnej oblasti (vpravo).

pri hľadaní na hranici prípustnej oblasti. Pri ostatných metódach priebeh začína v kladných hodnotách, pretože väčšina kandidátov najťažšie splniteľné ohraničenie porušuje (nižšie hodnoty porušovania v prípade opravnej procedúry sú dané tým, že okrem opravy vhodnosti jedincov dochádza aj k náhrade ich reprezentácií). K najrýchlejšiemu zotaveniu dochádza pri použití metódy sekvenčného spĺňania ohraničení – klesajúca časť reprezentuje tú fázu, keď sa metóda zameriava na zvyšovanie počtu tých jedincov, ktoré spĺňajú sledované druhé ohraničenie. Oproti iným metódam, pri viackriteriálnej optimalizácii dočasne nastáva nárast hodnoty a až po ňom rýchly pokles do záporných hodnôt. Je to spôsobené tým, že metóda neuvažuje jednotlivé ohraničenia oddelene ale ako celok – napriek tomu že hodnoty pre druhé ohraničenie stúpajú, splnenosť ohraničení ako celku sa zlepšuje.

Dosiahnutá vzdialenosť $o_2(a_\top)$ od nulovej hodnoty veľmi dobre koreš-ponduje s dosiahnutou hodnotou $\Phi(a_\top)$. Práve tie tri metódy, ktoré sa dokázali k hranici druhého ohraničenia priblížiť najmenej, skončili najhoršie aj pri tomto kritériu (pomerne vysoko nad hodnotou -3500) a javia sa ako nevhodné pre použitú testovaciu úlohu.

Matematické symboly

Symbol	Krátky popis
$\vec{\cdot}$	vektor
$\succ$	dominancia
$\lfloor . \rfloor$	zaokrúhlenie nadol
$\mid . \mid$	absolútna hodnota
$\langle \cdot \rangle$	stredná hodnota
$< \cdot, \cdot >$	interval
$[\cdot]$	vektor reprezentovaný zložkami
$\%$	modulo
$\cup$	zjednotenie
$\in$	prvok množiny
$\forall$	zovšeobecňovací kvantifikátor
$\exists$	existenčný kvantifikátor
α	parameter
$\mathbf{\Delta}$	odchýlka
ε	malé číslo
η	očakávaný počet rodičov
$\mathbf{\Phi}$	vhodnosť
$\mathbf{\Phi}_i$	i-ta (parciálna) vhodnosť
$\mathbf{\Phi}'$	zmenená (premapovaná) vhodnosť
$\overline{\mathbf{\Phi}}$	priemerná vhodnosť v populácii

Symbol	Krátky popis
$\Phi_\top$	maximálna vhodnosť v populácii
χ	náhodné číslo
μ	veľkosť populácie
π	Ludolfovo číslo (3,14159...)
Ψ	penalizácia
ϱ	veľkosť skupiny rodičov
$\sum$	súčet
θ	prah
a_i, $a_\top$	i-ty jedinec, najlepší jedinec
a'_i	modifikovaný jedinec
$a(i)$, $b(i)$	koeficienty
c	pomerový parameter
d	vzdialenosť
f	testovacia funkcia
g	funkcia
H	entropia
i, j, l	indexy
k	počet
K	konštanta
L	normalizačný člen
m	počet
$\mathcal{M}_s$	priestor prehľadávania
$\mathcal{M}_c^f$	priestor kandidátov (prípustná oblasť)
$\mathcal{M}_c^u$	priestor kandidátov (neprípustná oblasť)
MIN, MAX	minimálna a maximálna hodnota
n	počet
Nx	počet objektov typu "x"

Symbol	Krátky popis
o_i	i-te ohraničenie
$\overline{o_i}$	priemerná hodnota ohraničenia v populácii
p_c	pravdepodobnosť kríženia
p_{fm}	pravdepodobnosť nútenej mutácie
p_m	pravdepodobnosť mutácie
$p(.)$	pravdepodobnosť nejakého javu
P	populácia
r	polomer (oblasti, zdieľania)
r_i	i-te parciálne poradie
R	faktor redundancie
$\Re$	reálne čísla
s	riešenie
t	generácia
T	teplota
u_i, v_i	i-ta hodnota
w, w_i	váha, i-ta váha
x, y, z	premenné
X, Y, Z	body v priestore

Literatúra

[1] Beasley, D. – Bull, D.R. – Martin, R.R.: A Sequential Niche Technique for Multimodal Function Optimization. Evolutionary Computation, vol. 1, 1993, no. 2, 101–125.

[2] Cobb, H.G. – Grefenstette, J.J.: Genetic Algorithms for Tracking Changing Environments. In: Proc. of the 5th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1993, 523–530.

[3] Corne, D.W. – Knowles, J.D. – Oates, M.J.: The Pareto envelope-based selection algorithm for multiobjective optimization. In: Parallel Problem Solving from Nature - PPSN VI, Springer, LNCS 1917, 2000, 839–848.

[4] Deb, K.: Construction of test problems for multi-objective optimization. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO'99, Orlando, Florida (13-17 July 1999), 164–171.

[5] Deb, K.: Non-dominated sorting genetic algorithm. In: Multi-objective Optimization using Evolutionary Algorithms, John Wiley, 2001, 209–217.

[6] Deb, K. – Agrawal, S. – Pratap, A. – Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Parallel Problem Solving from Nature - PPSN VI, Springer, LNCS 1917, 2000, 849–858.

[7] Deb, K. – Goldberg, D.E.: An investigation of niche and species formation in genetic function optimization. In: Proc. of the 3rd Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1989, 40–50.

[8] Fitzgerald, A. – O'Donoghue, D.P.: Genetic Repair for Optimization under Constraints Inspired by Arabidopsis Thaliana. In: Parallel Problem Solving from Nature - PPSN X, Springer, LNCS 5199, 2008, 399–408.

[9] Fonseca, C.M. – Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proc. of the 5th Int. Conference on Genetic Algorithms, Morgan Kaufmann, 1993, 416–423.

[10] Fonseca, C.M. – Fleming, P.J.: Multiobjective optimization. In: Evolutionary Computation 2: Advanced Algorithms and Operators. IoP Publishing, Bristol, 2000, 25–37.

[11] Gan, J. – Warwick, K.: A Variable Radius Niching Technique for Speciation in Genetic Algorithms. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO-2000, Las Vegas, Nevada, (10-12 July, 2000), 96–103.

[12] Goldberg, D.E. – Wang, L.: Adaptive Niching via Coevolutionary Sharing. In: Genetic Algorithms and Evolution Strategies in Engineering and Computer Science, John Wiley & Sons, Chichester, 1998, 21–38.

[13] Grefenstette, J.J.: Genetic algorithms for changing environments. In: Parallel Problem Solving from Nature 2, North-Holland, 1992, 137–144.

[14] Hamida, S.B. – Schoenauer, M.: An Adaptive Algorithm for Constrained Optimization Problems. In: Parallel Problem Solving from Nature - PPSN VI, Springer, LNCS 1917, 2000, 529–538.

[15] Horn, J. – Nafpliotis, N. – Goldberg, D.E.: A niched pareto genetic algorithm for multiobjective optimization. In: Proc. of the IEEE World Congress on Computational Computation, Piscataway, NJ, 1994, 82–87.

[16] Jin, Y. – Branke, J.: Evolutionary Optimization in Uncertain Environments - A Survey. IEEE Trans. on Evolutionary Computation, vol. 9, 2005, no. 3, 303–317.

[17] Jin, Y. – Okabe, T. – Sendhoff, B.: Adapting weighted aggregation for multiobjective evolution strategies. In: First International Conference on Evolutionary Multi-criteria Optimization, Springer, LNCS 1993, 2001, 96–110.

[18] Kim, D.G. – Husbands, P.: Landscape Changes and the Performance of Mapping Based Constraint Handling Methods. In: Parallel Problem Solving from Nature - PPSN V, Springer, LNCS 1498, 1998, 221–230.

[19] Kirley, M. – Green, D.G.: An Empirical Investigation of Optimisation in Dynamic Environments Using the Cellular Genetic Algorithm. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO-2000, Las Vegas, Nevada (10-12 July, 2000), 11–18.

[20] Koziel, S. – Michalewicz, Z.: A Decoder-Based Evolutionary Algorithm for Constrained Optimization Problems. In: Parallel Problem Solving from Nature - PPSN V, Springer, LNCS 1498, 1998, 231–240.

[21] Leung, K-S. – Liang, Y.: Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO-2003, Chicago, (12-16 July, 2003), 1160–1171.

[22] Lewis, J. – Hart, E. – Ritchie, G.: A Comparison of Dominance Mechanisms and Simple Mutation on Non-stationary Problems. In: Parallel Problem Solving from Nature – PPSN V, Springer, LNCS 1498, 1998, 139–148.

[23] Mahfoud, S.W.: Niching methods for genetic algorithms. PhD dissertation, Univ. of Illinois, Urbana-Champaign, USA, 1995.

[24] Mahfoud, S.W.: A Comparison of Parallel and Sequential Niching Methods. In: Proc. of the 6th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, 136–143.

[25] Mahfoud, S.W.: Niching Methods. In: Evolutionary Computation 2: Advanced Algorithms and Operators. IoP Publishing, Bristol, 2000, 87–92.

[26] Mach, M.: Evolučné algoritmy – prvky a princípy. Elfa, Košice, 2009, ISBN 978-80-8086-123-0, 237 strán.

[27] Maneeratana, K. et al.: Compressed-objective genetic algorithm. In: Parallel Problem Solving from Nature – PPSN IX, Springer, LNCS 4193, 2006, 473–482.

[28] Mengshoel, O.J. – Goldberg, D.E.: Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement. In: Proc. of the Genetic

and Evolutionary Computation Conference GECCO-99, Orlando, Florida, (13-17 July, 1999), 409–416.

[29] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolutionary Programs (3rd edition), Springer, New York, 1996, 387 strán.

[30] Michalewicz, Z.: Constraint-preserving operators. In: Evolutionary Computation 2: Advanced Algorithms and Operators. IoP Publishing, Bristol, 2000, 62–68.

[31] Michalewicz, Z.: Repair algorithms. In: Evolutionary Computation 2: Advanced Algorithms and Operators. IoP Publishing, Bristol, 2000, 56–61.

[32] Mori, N. – Imanishi, S. – Kita, H. – Nishikawa, Y.: Adaptation to Changing Environments by Means of the Memory Based Thermodynamical Genetic Algorithm. In: Proc. of the 7th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1997, 299–306.

[33] Mori, N. – Kita, H. – Nishikawa, Y.: Adaptation to a Changing Environment by Means of the Feedback Thermodynamical Genetic Algorithm. In: Parallel Problem Solving from Nature – PPSN V, Springer, LNCS 1498, 1998, 149–157.

[34] Ng, K.P. – Wong, K.C.: A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. In: Proc. of the 6th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, 159–166.

[35] Ošmera, P.: An application of genetic algorithms with diploid chromosomes. In: Proc. of the 4th Int. Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks and Rough Sets, Brno, PC DIR, Czech Republic, 1998, 86–89.

[36] Petrowski, A.: A new selection operator dedicated to speciation. In: Proc. of the 7th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, 1997, 144–151.

[37] Ryan, C.: The degree of oneness. In: Proc. of the First Online Workshop on Soft Computing WSC1, 1996, 43–48.

[38] Ryan, C. – Collins, J.J.: Polygenic Inheritance – A Haploid Scheme that can Outperform Diploidy. In: Parallel Problem Solving from Nature – PPSN V, Springer, LNCS 1498, 1998, 178–187.

[39] Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proc. of the 1st Int. Conference on Genetic Algorithms, Lawrence Erlbaum Associates, New Jersey, 1985, 93–100.

[40] Schoenauer, M. – Michalewicz, Z.: Boundary operators for constrained parameter optimization problems. In: Proc. of the 7th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1997, 322–329.

[41] Schoenauer, M. – Xanthakis, S.: Constrained GA optimization. In: Proc. of the 5th Int. Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993, 573–590.

[42] Smith, A.C. – Coit, D.W.: Penalty functions. In: Evolutionary Computation 2: Advanced Algorithms and Operators. IoP Publishing, Bristol, 2000, 56–61.

[43] Ursem, R.K.: Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO-2000, Las Vegas, Nevada, (10-12 July, 2000), 19–26.

[44] Vemuri, V.R. – Cedeno, W.: Multi-niche crowding for multi-modal search. In: Practical Handbook of Genetic Algorithms, CRC Press, New York, 1995, 5–29.

[45] Yang, S.: On the Design of Diploid Genetic Algorithms for Problem Optimization in Dynamic Environments. In: Proc. of the IEEE Congress on Evolutionary Computation, Vancouver, BC, 2006, 1362–1369.

[46] Yang, S.: Explicit Memory Schemes for Evolutionary Algorithms in Dynamic Environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, Springer, Studies in Computational Intelligence, 2007, ISBN 978-3-540-49772-1, 3–28.

[47] Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. PhD Thesis, ETH Zurich, Switzerland, 1999.

[48] Zitzler, E. – Kunzli, S.: Indicator-based selection in multiobjective search. In: Parallel Problem Solving from Nature – PPSN VIII, Springer, LNCS 3242, 2004, 832–842.

[49] Zitzler, E. – Laumanns, M. – Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. TIK Report 103, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, 2001.

Register

Autor: Mach Marián
Názov: Evolučné algoritmy
Podnázov: Riešenie úloh
Vydanie: prvé
Náklad: 100 ks
Rozsah: 146 strán
Vydavateľ: Technická univerzita v Košiciach
Sadzba: LaTeX+ gnuplot
Formát: b5
Tlač: petit s.r.o., Košice
Rok: 2013

ISBN 978-80-553-1445-7