VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ MODEL S UČENÍM (LEM) PRO OPTIMALIZAČNÍ ÚLOHY

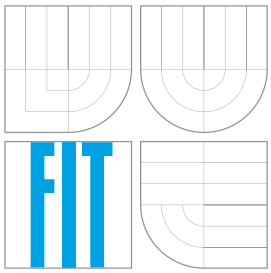
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL GRUNT

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ MODEL S UČENÍM (LEM) PRO OPTIMALIZAČNÍ ÚLOHY

LEARNABLE EVOLUTION MODEL FOR OPTIMIZATION (LEM)

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL GRUNT

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. JOSEF SCHWARZ, CSc.

BRNO 2014

Abstrakt

Tato práce se zabývá evolučním modelem s učením, relativně novou evoluční optimalizační metodou používající klasifikační algoritmy. Její optimalizační průběh je řízen dle charakteristiky rozdílu skupiny nejlepších od skupiny nejhorších řešení v populaci. Práce blíže představuje nové verze metody s klasifikačními algoritmy AdaBoost, SVM a také způsob využívání většího počtu skupin řešení. Kvality metod byly ověřovány na řadě experimentů ve statickém i dynamickém prostředí. Výsledky experimentů ukázaly, že metoda dosahuje nejlepších hodnot při menších velikostech skupin. Při srovnání s EDA (Estimation of Distribution Algorithm) optimalizačním algoritmem varianty evolučního modelu s učením dosahovaly srovnatelných a lepších výsledků rychleji. Celkově nejlépe si vedla varianta kombinující klasifikátory AdaBoost a SVM.

Abstract

My thesis is dealing with the Learnable Evolution Model (LEM), a new evolutionary method of optimization, which employs a classification algorithm. The optimization process is guided by a characteristics of differences between groups of high and low performance solutions in the population. In this thesis I introduce new variants of LEM using classification algorithm AdaBoost or SVM. The qualities of proposed LEM variants were validated in a series of experiments in static and dynamic enviroment. The results have shown that the metod has better results with smaller group sizes. When compared to the Estimation of Distribution Algorithm, the LEM variants achieve comparable or better values faster. However, the LEM variant which combined the AdaBoost approach with the SVM approach had the best overall performance

Klíčová slova

LEM, evoluční model s učením, evoluční algoritmus, strojové učení, klasifikace, AdaBoost, SVM, metoda podpůrných vektorů, optimalizace

Keywords

LEM, learnable evolution model, evolutionary algorithm, machine learning, classification, AdaBoost, SVM, support vector machines, optimization

Citace

Pavel Grunt: Evoluční model s učením (LEM) pro optimalizační úlohy, diplomová práce, Brno, FIT VUT v Brně, 2014

Evoluční model s učením (LEM) pro optimalizační úlohy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Josefa Schwarze, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Grunt
26. května 2014

Poděkování

Rád bych poděkoval panu doc. Ing. Josefu Schwarzovi, CSc. za rady, připomínky a náměty, které mi pomohly při tvorbě diplomové práce.

© Pavel Grunt, 2014.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **2**

2 Evoluční optimalizační techniky a metoda LEM **3**
2.1 Evoluční techniky . . . 3
2.2 Evoluční model s učením (LEM) . . . 4

3 Klasifikační metody použité v LEM **8**
3.1 AQ (Algorithm Quasi–Optimal) . . . 8
3.2 k–NN (k – nejbližších sousedů) . . . 9
3.3 ED (Entropy–based Discretization) . . . 10
3.4 Rozhodovací stromy . . . 10
3.5 PRISM . . . 11
3.6 Neuronová síť RCE . . . 12

4 Modifikace metody LEM **14**
4.1 LEM s klasifikátorem AdaBoost . . . 14
4.2 LEM s klasifikátorem SVM . . . 16
4.3 LEM se střídáním AdaBoost a SVM klasifikátoru . . . 17
4.4 Rozdělení populace do více skupin . . . 17
4.5 Evoluční fáze . . . 17
4.6 Odlišnosti od LEM3 . . . 19

5 Experimenty ve statickém prostředí **20**
5.1 Přehled používaných funkcí . . . 20
5.2 Velikost skupiny . . . 24
5.3 Více skupin . . . 28
5.4 Vliv evoluční části . . . 29
5.5 Srovnání LEM s původní LEM3 . . . 31
5.6 Porovnání metod LEM s EDA algoritmem . . . 33

6 Experimenty v dynamickém prostředí **40**
6.1 Pohybující se sinusový vrchol . . . 40
6.2 Úloha s více pohybujícími vrcholy . . . 43

7 Závěr **46**

A Obsah DVD **50**

B Manuál **51**

Kapitola 1

Úvod

Inženýrské úlohy jsou ve většině případů řešeny analytickými matematickými metodami. S rozvojem složitosti systémů však narůstá nejen časová náročnost řešení, mnohdy však již ani není možné danou úlohu řešit klasickou matematickou metodou. Nejen v těchto složitých případech se s výhodou využívají k řešení tzv. evoluční optimalizační algoritmy. Ty sebou přinášejí zejména jednoduchost a rychlost nalezení optimálního výsledku v přijatelném čase.

Relativně novou optimalizační metodu, jež přináší netradiční přístup v podobě zapojení technik z oblasti strojového učení do procesu optimalizace řešení úloh, představuje evoluční model s učením (Learnable Evolution Model – dále jen LEM).

Tato diplomová práce navazuje na můj semestrální projekt, jež seznamoval čtenáře s vývojem metody LEM (včetně použitých klasifikačních algoritmů) a s analýzou vlastností metody LEM z hlediska evolučního algoritmu. Diplomová práce představuje nejen nové verze využívající klasifikační algoritmy AdaBoost a SVM, ale i seznamuje a vysvětluje podstatu jejich úprav. Nedílnou součástí práce je i ověření vlastností a kvalit těchto metod ve statickém a dynamickém prostředí.

Kapitola 2

Evoluční optimalizační techniky a metoda LEM

V této kapitole jsou vysvětleny základní pojmy z oblasti evolučních výpočetních technik. Dále je popsán obecný princip evolučního algoritmu a blíže představena metoda LEM.

2.1 Evoluční techniky

Evoluční techniky představují rozmanitou rodinu výpočetních metod a postupů pro řešení optimalizačních problémů (úloh). Svým názvem odkazují na přímou analogii s evolučním vývojem organismů a jevů, které můžeme pozorovat v přírodě.

Základní pojmy, s nimiž evoluční techniky pracují, jsou:

- Jedinec (či chromozom) – Jedinec představuje jedno konkrétní řešení optimalizačního problému (úlohy). Obvykle se jedná o vektor čísel (binárních, celočíselných nebo reálných) představující hodnoty parametrů problému. Může být ale reprezentován pomocí nějaké komplexnější struktury (např. stromu). Jednotlivé složky jedince (chromosomu) se nazývají geny.
- Populace – Populace je množina jedinců. V některých evolučních algoritmech se pracuje i s populací o jednom jedinci.
- Evoluční operátor – Jedná se o nástroj pro tvorbu nových jedinců (potomků) z několika jiných jedinců (rodičů). Evoluční operátory nacházejí inspiraci v přírodních procesech (např. křížení, mutace). Mohou však být i specializovanější, což je případ metody LEM.
- Fitness funkce – Během evolučního procesu je nutné mít možnost porovnat kvalitu (fitness) jedinců. K tomu slouží fitness funkce, která jedince ohodnotí podle jeho reprezentace. Obecně by mělo platit, že čím je lepší jedinec (respektive řešení problému, které představuje), tím by pro něj měla být hodnota fitness funkce větší.

Přestože obecné schéma evolučního algoritmu působí velmi prostě (algoritmus 1), evolučně získaná řešení pomohla přijít s kvalitními a často neobvyklými řešeními rozličných problémů. Klíčovou roli nicméně vždy hraje člověk, který musí pro daný problém formulovat jakou reprezentaci má mít jedinec (s tím souvisí i případná definice evolučních operátorů), jak tuto reprezentaci ohodnotit (tj. definovat fitness funkci) a v neposlední řadě, jak získané řešení aplikovat.

Evoluční algoritmus vlastně nedělá nic jiného, než že vhodně prohledává prostor parametrů modelu problému. Není zaručeno, že řešení získaná tímto způsobem jsou optimální, ale bývají svou kvalitou optimu blízká a rychleji dosažitelná, než kdyby byl problém řešen analyticky.

Algoritmus 1 Základní schéma evolučního algoritmu:

Vytvoř počáteční populaci P
Ohodnoť populaci P fitness funkcí
while dokud není splněna ukončovací podmínka **do**
  Zvol vhodné rodiče z populace P
  Vytvoř populaci potomků P_c z vybraných rodičů evolučními operátory
  Ohodnoť populaci potomků P_c
  Vyber vhodné jedince z populace potomků P_c a nahraď jimi stejný počet jedinců z populace P
end while

2.2 Evoluční model s učením (LEM)

Optimalizační metoda LEM zjišťuje důvody, proč jsou někteří jedinci v populaci výrazně lepší než jiní. Tyto poznatky jsou poté použity k vytvoření nové generaci jedinců. Její techniky nejsou blízké přírodním jevům, tím se odlišuje od klasických evolučních algoritmů.

2.2.1 LEM1

Evoluční model s učením byl poprvé popsán Michalskim v roce 1998. Jeho práce [1] se zabývá metodou LEM z hlediska kombinace technik strojového učení a evolučních algoritmů.

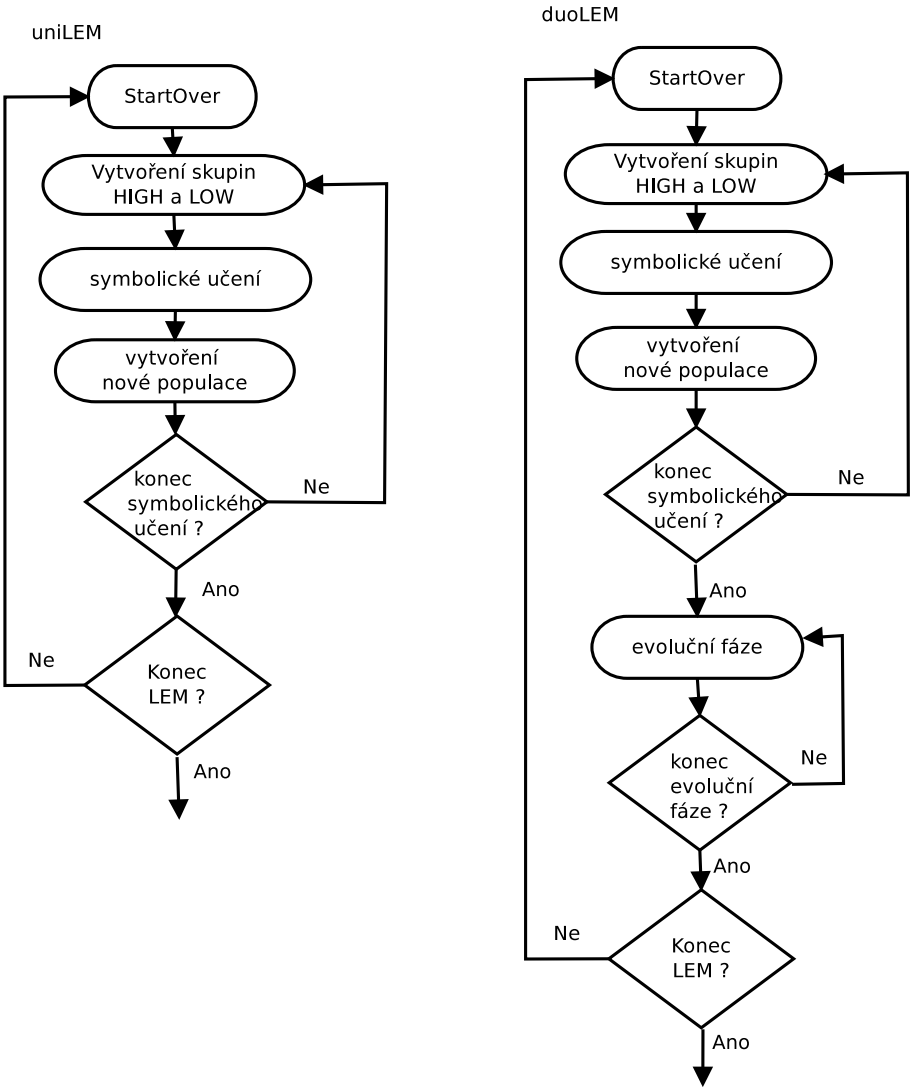
Princip metody je jednoduchý. V každé generaci je dle hodnoty fitness funkce vybrána skupina nejlepších jedinců (HIGH) a nejhorších jedinců (LOW). Pomocí strojového učení se získá charakteristika rozdílu skupiny HIGH oproti skupině LOW, následně se podle ní vytvářejí noví jedinci. Pokud po určitou dobu nedochází ke zlepšení fitness populace, fáze učení je vystřídána evoluční fází.

Algoritmus 2 Pseudokód LEM1:

vytvoř počáteční populaci
while dokud není splněna ukončovací podmínka **do**
  while dochází ke zlepšení řešení **do**
    Prováděj evoluční fázi běžným genetickým algoritmem
  end while
  while dochází ke zlepšení řešení **do**
    Na základě fitness urči skupiny jedinců HIGH a LOW
    Použij metodu strojového učení pro charakterizaci rozdílů mezi skupinami HIGH a LOW jedinců
    Generuj novou populaci jedinců pomocí popisu skupiny HIGH, jež nahradí jedince nepatřící do HIGH
  end while
end while

2.2.2 LEM2

Metoda LEM2 byla představena v práci Cervoneho [2] už v roce 1999. Významnou změnou oproti LEM1 je možnost úplně vynechat evoluční fázi. Tato varianta je nazvána uniLEM, původní se nazývá duoLEM. Vývojový diagram těchto variant je na obrázku 2.1.



Obrázek 2.1: varianty metody LEM2

Popis jednotlivých fází LEM2 dle [3]

- varianty operace StartOver – generování počáteční populace:
  - Random – populace náhodně vygenerovaných jedinců
  - Select-elite – náhodně vygenerovaného jedince vlož do populace, je-li jeho fitness lepší než určitý práh
  - Avoid post-failures – náhodně vygenerovaného jedince vlož do populace, nevyhovuje-li popisu skupiny LOW

  - Use-recommendations – tato varianta používá k vytvoření populace popis(y) skupin(y) HIGH, po nichž nastalo výrazné zlepšení fitness
  - Generate a variant – modifikace poslední populace určitým způsobem (např. mutací jedinců)

- způsoby vytvoření skupin HIGH a LOW:
  - fitness based – zvolí se prahy fitness pro skupiny HIGH a LOW. Je-li fitness jedince nižší než práh LOW, jedinec bude patřit do skupiny LOW. Má-li jedinec fitness vyšší než práh HIGH, patří do skupiny HIGH. Nevýhodou tohoto způsobu výběru jedinců je to, že může vytvářet velikostně nevyvážené skupiny HIGH a LOW.
  - population based – zvolí se prahy, které určí, kolik jedinců připadne do které skupiny (např. třetina s nejlepší fitness připadne do HIGH, třetina s nejhorší fitness do skupiny LOW).

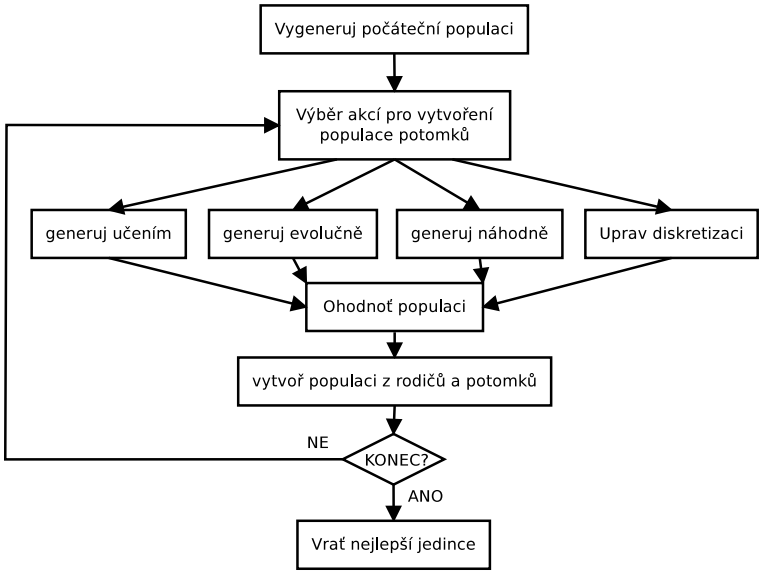
- varianty generování nové populace:
  - Bez historie – nová populace je vytvořena dle popisu skupiny HIGH.
  - Population lookback – při fázi symbolického učení se bere do úvahy několik posledních skupin LOW.
  - HIGH-group description lookback – jedinci jsou generováni na základě popisu skupiny HIGH, ale jsou vloženi do populace, pokud vyhovují i několika předchozím popisům skupiny HIGH.
  - LOW-group description lookback – jedinci se generují dle popisu skupiny HIGH, jsou vloženi do populace, pokud nevyhovují popisu skupiny LOW
  - Incremental specialization – popis skupiny HIGH se vytváří z popisu předchozí HIGH skupiny a popisu aktuální generace. Popis HIGH se tak více specializuje.

2.2.3 LEM3

V roce 2005 Janusz Wojtusiak a Ryszard S. Michalski představují novou verzi LEM [4]. Metoda LEM3 vychází z metody LEM2, ale spíše než o její úpravu se jedná o novou formulaci dřívějších postupů.

Zatímco v předchozích verzích LEM docházelo ke střídání evoluční fáze a fáze učení, LEM3 přináší přístup, kdy nová generace jedinců vzniká z původní částečně aplikací strojového učení, částečně evolučním algoritmem a částečně vygenerováním úplně nových jedinců. O tom, kolik jedinců vznikne kterým způsobem, rozhoduje tzv. akční profilová funkce. Vývojový diagram je na obrázku 2.2.

Významným prvkem metody je použití metody ANCHOR [5] pro převod reálných hodnot na celočíselné.



Obrázek 2.2: Vývojový diagram metody LEM3

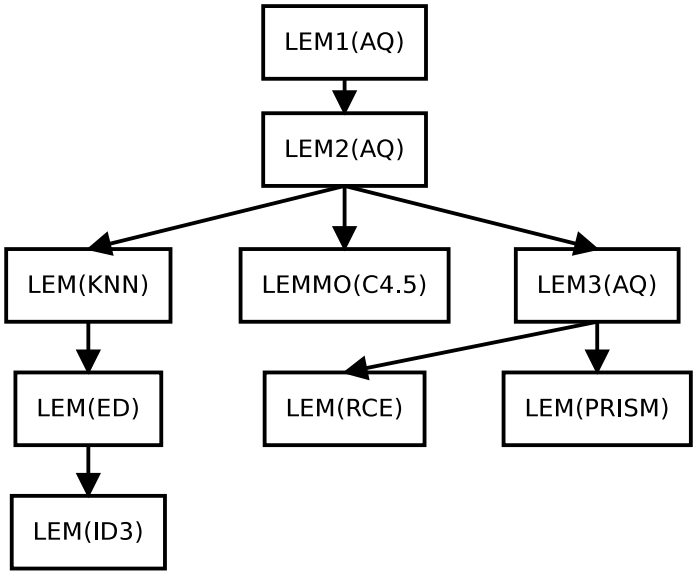
Popis jednotlivých fází LEM3

- Generování počáteční populace – Nejdříve se provede diskretizace prostoru pro každou dimenzi. Následně se náhodně vygeneruje a ohodnotí požadovaný počet jedinců.
- Výběr akcí pro vytvoření populace potomků – Řídící mechanismus, který rozhoduje, zda není potřeba upravit diskretizaci a kolik jedinců má být vygenerováno kterým způsobem. Dále umožňuje restartovat celý proces, pokud dlouhodobě nedochází ke zlepšení hodnoty fitness.
  - Generuj učením – Z populace se dle hodnoty fitness vybere skupina nejhorších a nejlepších jedinců. Tento soubor jedinců slouží jako trénovací množina klasifikačního algoritmu. Dle výsledku klasifikace se vytvářejí noví jedinci.
  - Generuj evolučně – Pomocí ruletového kola jsou vybráni dva reprezentanti, mezi nimiž proběhne jednobodové křížení. To se opakuje, dokud není vygenerován požadovaný počet jedinců.
  - Generuj náhodně – Určitý počet jedinců je vygenerován zcela náhodně a vložen do populace potomků.
  - Uprav diskretizaci – Pokud po několika generacích nedochází ke zlepšení fitness, zvýší se diskretizace v „okolí“ nejlepších jedinců.
- Ohodnoť populaci – Pro všechny jedince v populaci potomků se vyhodnotí fitness.
- Vytvoř populaci z rodičů a potomků – Vznikne nová populace o velikosti rodičovské. Výběr jedinců, kteří zůstanou, může být dle hodnoty fitness, turnajem apod.

Kapitola 3

Klasifikační metody použité v LEM

Klasifikační algoritmus je „základním kamenem“ metody LEM. Jeho změna často vyžaduje provedení výrazných úprav metody. V této kapitole je uveden seznam klasifikačních metod s jejich stručným popisem a způsobem vytváření nových jedinců. Dále je u každé z metod uveden způsob vytváření nových jedinců a optimalizační úloha, v níž byla daná varianta metody LEM použita.



Obrázek 3.1: Vzájemný vztah mezi různými variantami metody LEM

3.1 AQ (Algorithm Quasi–Optimal)

Klasifikační algoritmus AQ byl použitý v metodě LEM jako první. Patří mezi algoritmy induktivního učení. Byl výhradně využíván ve verzích LEM, jejichž autorem byl Michalski nebo jeho kolegové.

Algoritmus AQ vytváří soubor pravidel (hypotézy), která pokrývají všechny reprezentanty jedné skupiny a současně nepokrývají všechny reprezentanty druhé skupiny (viz algoritmus 3). Stěžejní částí algoritmu je generování tzv. *star* – souboru pravidel, která pokrývají jednoho reprezentanta e^+ první skupiny, ale nepokrývají žádného reprezentanta druhé skupiny.

Algoritmus 3 Pseudokód algoritmu AQ (dle [4]):

HYPOTEZY = $\emptyset$
while HYPOTEZY nepokrývají všechny reprezentanty skupiny HIGH **do**
  Vyber nepokrytého jedince e^+ skupiny HIGH
  Vytvoř *star* G(e^+, skupina LOW)
  Přidej do množiny HYPOTEZY nejlepší pravidlo ze *star*
end while

LEM s AQ byl použit v různých oblastech. Od základního testování při funkční optimalizaci, hledání optimálních parametrů číslicového filtru [6]. V praxi pak pro návrh tepelných výměníků [7]. Zajímavá byla i aplikace pro evoluční návrh hardwaru [8].

3.1.1 Generování jedinců

Výstupem algoritmu AQ je soubor pravidel/intervalů, v nichž se nachází jedinci skupiny HIGH. Ve verzi LEM1 [1] se používalo pouze jedno pravidlo ke generování jedinců. V případě LEM3 [4] jsou definovány tři způsoby tvorby jedinců:

- Náhodně – Pro každé pravidlo urči, kolik jedinců (potomků) se jím má vytvořit. Pro každý gen vytvářeného potomka náhodně vygeneruj hodnotu vyhovující odpovídajícímu pravidlu. Jestliže není některý gen pokryt pravidle, zvol libovolného jedince z populace a použij hodnotu jeho příslušného genu.
- Dle fitness – Náhodně vyber jedince (pravděpodobnost výběru je závislá na fitness) a urči seznam pravidel, kterým vyhovuje. Náhodně z nich zvol jedno pravidlo (s pravděpodobností výběru dle jeho významnosti) a pro každý gen, který pokrývá, vygeneruj hodnotu, která mu vyhovuje. Pro geny, které nejsou pokryty pravidlem, použij hodnoty vybraného jedince.
- Rodič dle pravidla – Pro každé pravidlo urči, kolik potomků má být dle něj vytvořeno. Každému potomku vyber rodiče (tj. jedince, který vyhovuje pravidlu). Hodnoty genů potomka vytvoř dle pravidla. Pro ty geny, které nejsou pokryty, použij hodnoty odpovídajícího genu rodiče.

3.2 *k*–NN (*k* – nejbližších sousedů)

Algoritmus k–NN patří mezi nejjednodušší klasifikační metody. Veškerá činnost algoritmu se odkládá až na fázi klasifikace nových dat. Při klasifikaci nového vzorku spočítá jeho vzdálenost od reprezentantů trénovací množiny a určí se tak množina k nejbližších sousedů. Novému vzorku je pak predikována třída, jakou má většina z jeho k nejbližších sousedů.

Algoritmus k–NN byl jako jádro fáze učení metody LEM vyzkoušen na problémech funkční optimalizace [9].

3.2.1 Generování jedinců

Nejdříve se dle fitness určí skupiny nejlepších jedinců (HIGH) a nejhorších jedinců (LOW). Následně se generuje noví jedinec pomocí klasických evolučních operátorů křížení a mutace. Tomu to jedinci je určena třída dle k–NN, odpovídá-li skupině HIGH, jedinec je schválen, jinak je zahozen. To se opakuje, dokud není schválen požadovaný počet jedinců.

3.3 ED (Entropy–based Discretization)

Informační entropie je míra značící množství informace obsažené v rámci určité zprávy [10]. Při klasifikaci se využívá pro určení, který parametr a jeho hodnota má největší vliv na informační zisk. V kombinaci s metodou LEM byla ED použita na problém funkční optimalizace [11].

3.3.1 Generování jedinců

Obvyklým způsobem se určí skupiny jedinců HIGH a LOW. Následně se pro každý gen určí tzv. *cut point*, bod který dělí prostor na dvě části, v každé z nich má převahu jedna ze skupin HIGH a LOW. Noví jedinci se náhodně vytvoří v podprostorech s převahou skupiny HIGH.

3.4 Rozhodovací stromy

Jako jádro fáze učení metody LEM byl použit algoritmus pro tvorbu rozhodovacího stromu ID3 i jeho vylepšení C4.5. Oba algoritmy patří mezi často používané metody strojového učení. Liší se způsobem výběru atributu jako kořene rozhodovacího stromu. Algoritmus ID3 zvolí za kořen atribut s největším informačním ziskem, C4.5 používá normalizovanou hodnotu informačního zisku. Po volbě kořene se algoritmus rekurzivně zavolá na každou z podmnožin vzniklých dělením dle použitého atribut. Rozhodovací strom vznikne, jakmile jsou podmnožiny prázdné (viz algoritmus 4). Před použitím algoritmu je nutné převést spojité hodnoty na diskrétní.

V kombinaci s LEM byl algoritmus ID3 využit při vytváření univerzálního klasifikátoru [12], ale i pro funkční optimalizaci [13]. V případě algoritmu C4.5 se jednalo o použití při optimalizaci vodovodní sítě [14].

Algoritmus 4 funkce *VytvořStrom*(množina vzorů S, seznam atributů L) (dle [10]):

vytvoř nový uzel N
if všechny vzory z S jsou ve stejné třídě C **then**
  return N jako list dané třídy C
end if
if $L = \emptyset$ **then**
  return N jako list nejběžnější třídy ve množině S
end if
Vyjmi *vhodný* atribut A ze seznamu L
Pojmenuj uzel N jménem atributu A
for každou možnou hodnotu a_i atributu A **do**
  Vytvoř větev z uzlu N pro podmínku $A = a_i$
  Nechť $s_i = \{s | s \in S$ pro něž platí $A = a_i\}$
  if $s_i = \emptyset$ **then**
    připoj k větvi list s nejběžnější třídou v množině S
  else
    připoj k větvi strom vzniklý voláním *VytvořStrom*(s_i, L)
  end if
end for

3.4.1 Generování jedinců

Průchodem rozhodovacím stromem od kořene k listu, který znamená klasifikaci do skupiny HIGH, získáme soubor pravidel (intervalů) popisující jedince skupiny HIGH. Dle těchto pravidel je možné generovat jedince obdobně jako u algoritmu AQ.

3.5 PRISM

Algoritmus PRISM pracuje na podobném principu jako AQ. Vytváří pravidla, která popisují jednu třídu (např. skupinu HIGH), na rozdíl od AQ při vytváření pravidel nebere do úvahy ostatní třídy (skupina LOW). To může vést k tomu, že výsledná pravidla budou příliš obecná.

Algoritmus PRISM byl vyzkoušen na problémech funkční optimalizace v diplomové práci Martina Weisse [15].

Algoritmus 5 PRISM (dle [15]):

for každou třídu C **do**
  Inicializuj množinu E prvky z trénovací množiny
  while E obsahuje prvky patřící do třídy C **do**
    Vytvoř obecné pravidlo $R = (\textbf{if}$ true $\textbf{then}\ C)$
    while nemůže být R konkrétnější **do**
      nalezni pár (atribut A, hodnota v) s maximální pravděpodobností $p(C|S_\alpha)$,
      $S_\alpha = \{s|s \in E$ a mají hodnotu atributu $A = v\}$
      přidej do pravidla R podmínku $(\textbf{if}\, A = v\ \textbf{then}\ C)$
    end while
    odstraň z E všechny prvky pokryté pravidlem R
  end while
end for

3.5.1 Generování jedinců

Jelikož je výstupem soubor pravidel, generování nových jedinců probíhá stejným způsobem jako u algoritmu AQ.

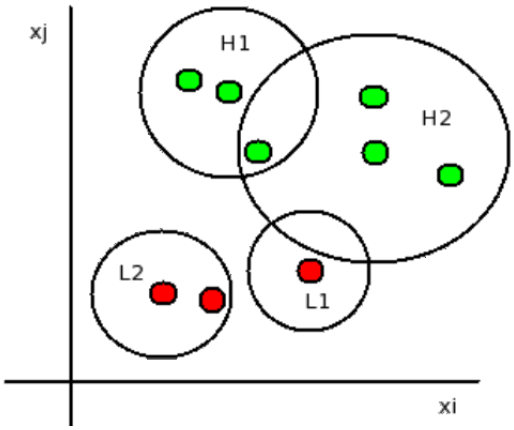
3.6 Neuronová síť RCE

Zajímavou aplikací bylo použití neuronové sítě RCE (Restricted Coulomb Energy) jako klasifikátoru. Neuronová síť RCE při trénování (algoritmus 6) pracuje s neurony uspořádanými do několika vrstev. Neurony jsou přidávány nebo je jim měněna velikost poloměru tak, aby pokryly trénovací sadu. Každý neuron je reprezentován vahou, která odpovídá souřadnicím v prostoru a poloměrem [15]. Příklad natrénované sítě je na obrázku 3.2.

Algoritmus 6 Trénování neuronové sítě RCE (dle [15]):

Na počátku jsou všechny vrstvy prázdné
Nastav proměnné *modif* a *hit* na false
repeat
  $hit \leftarrow$ false
  for každý vstupní vektor x z trénovací množiny **do**
    for každý neuron n skryté vrstvy **do**
      Vypočítej vzdálenost d mezi n a x
      if $d \leq$ poloměr neuronu n **then**
        if třída neuronu n je shodná se třídou vzoru x **then**
          $hit \leftarrow$ true
        else
          Zmenši poloměr n, aby byl menší než d
          $modif \leftarrow$ true
        end if
      end if
    end for
    if $hit =$ false **then**
      Vytvoř ve skryté vrstvě nový neuron n_h na souřadnicích vstupního vektoru x a nastav mu maximální možnou hodnotu poloměru
      $modif \leftarrow$ true
      if ve výstupní vrstvě existuje neuron pro třídu vektoru x **then**
        Připoj k němu neuron n_h
      else
        Vytvoř nový výstupní neuron pro třídu vstupního vektoru x a připoj k němu neuron n_h
      end if
    end if
  end for
until $modif = false$

Použití neuronové sítě RCE bylo navrženo a vyzkoušeno na problémech funkční optimalizace v diplomové práci Martina Weisse [15].



Obrázek 3.2: Natrénovaná neuronová síť RCE se čtyřmi neurony (převzato z [15])

3.6.1 Generování jedinců

Jak je zobrazeno na obrázku 3.2, neurony mají definovanou pozici a poloměr v prostoru. Nejjednodušším způsobem je vybrat některý z neuronů skupiny HIGH a náhodně vygenerovat jedince, který bude uvnitř prostoru definovaným zvoleným neuronem (hyperkoulí).

Jelikož je nutné zařídit, aby nedocházelo k přesahu mimo prostor hyperkoule, je generování založeno na náhodném určení souřadnice pro první dimenzi (tj. x_1, respektive její vzdálenosti R), zbylé body se dopočítají při znalosti úhlu mezi dimenzemi následovně [15]:

$$R \quad \ldots \quad \text{náhodně zvolená vzdálenost nového jedince} \tag{3.1}$$

$$x_k \quad = \quad R \cos \phi_k \prod_{i=1}^{k-1} \sin \phi_i \tag{3.2}$$

Kapitola 4

Modifikace metody LEM

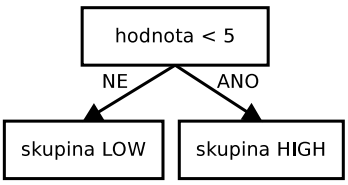
V této kapitole představím modifikace metody LEM, které jsem provedl především za účelem rychlejšího se přiblížení k hledanému optimu.

Jak bylo uvedeno v kapitole 3, vznikla řada různých modifikací metody LEM pomocí změny klasifikačního algoritmu. Některé z nich si vedly velmi dobře, i když používaly spíše jednodušší klasifikační algoritmy. Rovněž způsoby tvorby nových jedinců často nebyly příliš propracované. Proto jsem se rozhodl vytvořit modifikace metody LEM3, které budou používat pokročilejší klasifikační algoritmy, konkrétně AdaBoost a SVM (verze LEM budou dále nazývány LEM(AdaBoost) a LEM(SVM)). Ovšem na rozdíl od LEM3 tyto modifikace nepoužívají žádnou metodu pro diskretizaci reálných hodnot.

4.1 LEM s klasifikátorem AdaBoost

Klasifikační algoritmus AdaBoost pracuje na principu skládání jednoduchých klasifikátorů do jednoho silného. Jedinou podmínkou je, aby chyba klasifikátoru na trénovací sadě byla menší než 0,5. O principu algoritmu AdaBoost lze zjednodušeně možné říct, že ke každému z T klasifikátoru h_t hledá hodnotu α_t takovou, aby chyba err výsledného klasifikátoru $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$ byla nejmenší.

Jako dostatečný klasifikátor poslouží i jednoúrovňový rozhodovací strom (tzv. rozhodovací pařez – obrázek 4.1), který rozhoduje pouze podle jedné z dimenzí řešení a zvolené hodnoty prahu. Algoritmus 7 popisuje, jak se naleznou nejvhodnější klasifikátory.



Obrázek 4.1: Rozhodovací pařez

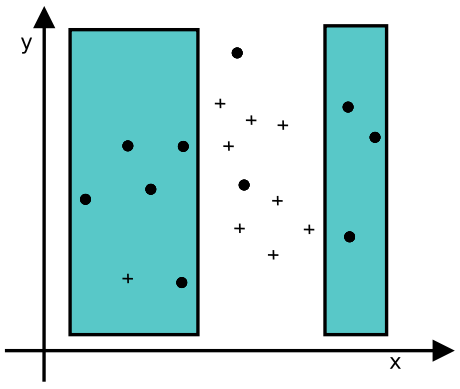
Algoritmus 7 Pseudokód algoritmu AdaBoost:

Každému z N vzorů (X_i, y_i) trénovací množiny přiřaď váhu $d_i \leftarrow 1/N$
for $j = 1 \ldots T$ **do**
  najdi klasifikátor K_j s nejmenší váženou chybou $\epsilon_j \leftarrow \sum_{i=1}^{N} d_i \cdot (y_i \neq K_j(X_i))$
  if $\epsilon_j > 0{,}5$ **then**
    skonči
  end if
  ulož klasifikátor K_j s vahou $\alpha_j \leftarrow 0.5 \cdot \log((1 - \epsilon_j)/\epsilon_j)$
  $Z_j \leftarrow 0$
  for $i = 0 \ldots N$ **do**
    if $y_i = K_j(X_i)$ **then**
      $d_i \leftarrow d_i \cdot (1 - \epsilon_j)/\epsilon_j)$
    end if
    $Z_j \leftarrow Z_j + d_i$
  end for
  for $i = 0 \ldots N$ **do**
    $d_i \leftarrow d_i / Z_j$
  end for
end for

4.1.1 Vytváření nových jedinců dle AdaBoost klasifikátoru

AdaBoost klasifikátor vybere dle algoritmu 7 dostatečný soubor jednoúrovňových rozhodovacích stromů, které od sebe odliší dvě zadané skupiny jedinců. Každý z těchto stromů nese informaci o dimenzi a prahu, na základě nichž dělí celý definiční obor na dvě části. Pokud pro jednu dimenzi je použito více stromů, získáme intervaly, v nichž mají převahu jedinci jedné ze skupin.

Při tvorbě nových jedinců tedy pro každou z dimenzí získáme množinu intervalů, z nichž si náhodně jeden vybereme a vygenerujeme hodnotu, která do něj náleží. Pokud pro některou dimenzi (gen jedince) není definovaný žádný interval, pak zkopírujeme hodnotu příslušné dimenze od libovolného jedince v populaci.



Obrázek 4.2: Příklad vhodných oblastí dle AdaBoost klasifikátoru

4.2 LEM s klasifikátorem SVM

Metoda podpůrných vektorů (Support Vector Machines) představuje odlišný způsob klasifikace, než jaký byl použit při předchozích variantách metody LEM. Zatímco ostatní klasifikační algoritmy vytvářely abstraktní model z trénovacích dat, který byl následně použit pro klasifikaci, SVM pro klasifikaci přímo využívá jednotlivé prvky trénovací množiny.

Základní varianta SVM vytváří lineární klasifikátor, jehož podstatou je nalezení co nejširšího pásu, jež by oddělil prvky rozdílných tříd trénovací množiny. Hraniční přímky tohoto pásu jsou definovány některými z prvků trénovací množiny. Tyto prvky pak nazýváme podpůrné vektory. Jelikož tato varianta SVM v podstatě jen rozděluje definiční obor na dva podobory, není vhodná pro použití v LEM. Je totiž možné, že trénovací vzory z různých tříd budou příliš podobné, pak je obtížné najít dělící pás, který má malou chybu. Ale i v případě nalezení tohoto pásu je klasifikace do tříd založena na přímce, tedy v případě LEM by popis skupiny jedinců byl příliš obecný. Tento problém je známý a existuje technika pro jeho řešení. Touto technikou je tzv. jádrová transformace, která umožňuje zvýšit dimenzionalitu dat tak, že jsou od sebe lépe oddělitelná. Dělících pásů po provedení jádrové transformace je pak i několik různého tvaru.

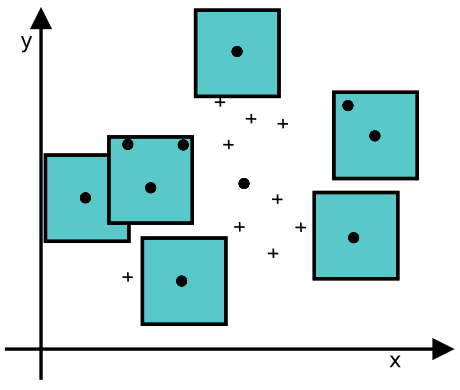
Jádrových transformací existuje celá řada. Rozhodl jsem se použít transformaci pomocí radiální bázové funkce. Prostřednictvím ní se počítá rozdíl dvou vektorů následovně:

$K(U,V) = e^{(-\gamma \cdot ||U-V||)}$, γ je konstanta ovlivňující oblast vzájemného působení vektorů

Časově nejnáročnější částí při trénování SVM klasifikátoru je nalezení podpůrných vektorů. Cílem je totiž maximalizace velikosti dělícího pásu při minimalizaci chyby klasifikace. Tento problém je řešitelný pomocí techniky SMO[16] (Sequential Minimal Optimization), která byla navržena speciálně pro rychlé trénování SVM klasifikátoru.

4.2.1 Vytváření nových jedinců dle SVM klasifikátoru

Generování jedinců je založeno na podpůrných vektorech. Nový jedinec vznikne v oblasti vlivu náhodně zvoleného podpůrného vektoru tak, že hodnota jeho genu je součtem odpovídající hodnoty dimenze podpůrného vektoru a náhodné hodnoty z intervalu $< -r, r >$, kde $r = \sqrt{0{,}5/\gamma}$.



Obrázek 4.3: Příklad vhodných oblastí dle SVM klasifikátoru

4.3 LEM se střídáním AdaBoost a SVM klasifikátoru

Jak ukazují obrázky 4.2 a 4.3, pravidla pro vytváření jedinců dle klasifikátoru se výrazně liší. Pravidla z AdaBoost klasifikátor se snaží pokrýt skupinu jedinců jako celek, jsou více obecná než oblasti vymezené SVM klasifikátorem, které se vztahují přímo k jedincům použitým k trénování.

Jelikož se metody LEM(AdaBoost) a LEM(SVM) odlišují pouze při tvorbě jedinců, spojení obou metod není výrazně problematické. Střídání klasifikačních algoritmů se bude provádět každou generaci.

Kombinace obou přístupů by měla pomoci vyrovnat rozdíly mezi LEM(AdaBoost) a LEM(SVM). Jak je tento způsob úspěšný je ukázáno při porovnání s EDA algoritmem (podkapitola 5.6).

4.4 Rozdělení populace do více skupin

V dřívějších verzích LEM a jejích modifikacích se z populace vybraly pouze dvě skupiny. Při používání pouze dvou výkonnostních skupin, nejlepších a nejhorších jedinců, se část populace nevyužívá pro vytváření popisu. Jedinci se vytváří pouze podle jednoho popisu. Je-li tento popis příliš obecný (obvykle při skupinách s větším počtem jedinců), nedochází k vytváření kvalitních jedinců. Naopak je-li příliš konkrétní, může se stát, že populace zcela ztratí diverzitu.

Při použití tří skupin už získáváme tři popisy k vytváření jedinců (klasifikátor se trénuje vždy ze dvou skupin, v tomto případě je možné vytvořit tři dvojice). Obecně pro n skupin je to $\binom{n}{2}$ popisů. Z většího počtu skupin o menším počtu jedinců tedy dostaneme několik specifičtějších popisů.

Algoritmus 8 Vytváření jedinců podle více skupin:

Vytvoř N skupin G, střídavě o k nejlepších/nejhorších nevybraných jedincích. Seřaď skupiny podle fitness od nejlepší po nejhorší.
for $i = 1 \dots N$ **do**
  for $j = i + 1 \dots N$ **do**
    Trénuj klasifikátor K, jako pozitivní vzory slouží jedinci skupiny G_i, negativní vzory jsou jedinci skupiny G_j.
    Získej z klasifikátoru K pravidla popisující jedince skupiny G_i.
    Dle získaných pravidel vytvoř příslušný počet nových jedinců
  end for
end for

Využívání více skupin by mohlo pomoci při uváznutí v lokálních extrémech díky prozkoumání většího počtu oblastí.

4.5 Evoluční fáze

Ve všech předchozích verzích metody LEM byl nějakým způsobem zapojen evoluční algoritmus. Bylo tomu tak proto, aby populace neztratila rozmanitost a aby byla vhodně kombinována řešení generovaná dle klasifikátoru.

Navíc v pozdější fázi evolučního procesu mohou být pravidla popisující nejlepší jedince příliš konkrétní. V těchto případech mohou klasické evoluční operátory pomoci zlepšit dosažené řešení.

Zvolil jsem kombinaci tří evolučních operátorů, z nichž každý přidává do evolučního procesu jiné vlastnosti. Uniformní křížení (algoritmus 9) vytváří nového jedince kombinací hodnot genů dvou jedinců. Aritmetické křížení (algoritmus 10) provádí vážený součet genů rodičů a mutace (algoritmus 11) změní hodnoty některých genů jedince. Obdobná kombinace evolučních operátorů byla použita v LEM2 [2].

Algoritmus 9 Uniformní křížení:

Vytvoř geny potomka g^P z genů rodičů g^{R_1} a g^{R_2} s pomocí rovnoměrného rozdělení U následovně
for $i = 1 \ldots |g^{R_1}|$ **do**
  if $U(0{,}0, 1{,}0) > 0{,}5$ **then**
    $g_i^P \leftarrow g_i^{R_1}$
  else
    $g_i^P \leftarrow g_i^{R_2}$
  end if
end for

Algoritmus 10 Aritmetické křížení:

Vytvoř geny potomka g^P z genů rodičů g^{R_1} a g^{R_2} s pomocí rovnoměrného rozdělení U následovně
for $i = 1 \ldots |g^{R_1}|$ **do**
  $a_i \leftarrow U(-0{,}25, 1{,}25)$
  $g_i^P \leftarrow a_i \cdot g_i^{R_1} + (1 - a_i) \cdot g_i^{R_2}$
end for

Algoritmus 11 Mutace:

Vytvoř geny potomka g^P z genů rodiče g^R s pomocí pravděpodobnosti p, rovnoměrného rozdělení U a normálního rozdělení N se směrodatnou odchylkou σ následovně
for $i = 1 \ldots |g^R|$ **do**
  if $U(0, 1) < p$ **then**
    $g_i^P \leftarrow g_i^R + N(0, \sigma)$
  else
    $g_i^P \leftarrow g_i^R$
  end if
end for

4.6 Odlišnosti od LEM3

LEM(AdaBoost) i LEM(SVM) se od sebe odlišují pouze v použitém klasifikačním algoritmu, jinak je jejich činnost zcela shodná. Jedinec představuje vektor reálných čísel. Tato podoba je použita po celou dobu běhu algoritmu, a to jak pro klasifikaci i při aplikaci evolučních operátorů.

Naproti tomu LEM3 primárně používá reprezentaci vektorem nezáporných čísel, která jsou mapována na reálné hodnoty diskretizační metodou ANCHOR. Metoda ANCHOR celkově hraje důležitou roli v celém procesu, pevně totiž určuje množinu reálných hodnot, kterých mohou geny jedince dosahovat. S tím souvisí kroky potřebné pro úpravu této množiny v průběhu evolučního procesu.

Jelikož navržené modifikace žádnou diskretizační metodu nepoužívají, je jejich jeden krok výrazně jednodušší (algoritmus 12). Metoda LEM3 se spoléhá na časté restarty k získání lepších jedinců – hodnoty maximálního počtu mutací MAX_M jsou v řádu jednotek. U LEM(AdaBoost) a LEM(SVM) je lepší používat hodnoty v řádu desítek. Počet generací bez zlepšení MAX_G by měl být menší než deset.

Algoritmus 12 krok metody LEM:

if nebyla dosažená lepší fitness **then**
  $G_{cnt} \leftarrow G_{cnt} + 1$
  if $G_{cnt} > MAX_G$ **then**
    $G_{cnt} \leftarrow 0$
    if $M_{cnt} < MAX_M$ **then**
      $M_{cnt} \leftarrow M_{cnt} + 1$
      Vytvoř populaci potomků P_c mutací každého jedince z populace rodičů P
    else
      Restartuj metodu
    end if
  end if
else
  $G_{cnt} \leftarrow 0$
  Vytvoř populaci potomků P_c z populace rodičů P částečně pomocí pravidel z klasifikátoru, evolučních operátorů a částečně zcela náhodně
end if
Ohodnoť populaci potomků P_c
Novou populaci P tvoří $|P|$ nejlepších jedinců z $P \cup P_c$

Kapitola 5

Experimenty ve statickém prostředí

V této kapitole jsou popsány experimenty, které byly prováděny za účelem ověření vlastností modifikací metody LEM a zjištění vlivu některých parametrů na optimalizační průběh. Dále jsou modifikace porovnávány s původní implementací LEM3 a s EDA algoritmem.

Metody LEM(AdaBoost) a LEM(SVM) byly implementovány v jazyce C++ dle standardu C++11. Veškeré experimenty proběhly v prostředí 64 bitového operačního systému Xubuntu 12.10 na stroji s procesorem Intel® Pentium® Dual CPU T3400 @ 2.16GHz.

Při řešení problému minimalizace funkce f se pracuje s fitness funkcí definovanou jako $fitness(X) = C - f(X)$, kde C je hodnota větší než největší dosažená hodnota funkce f během evoluce. Hodnoty uvedené v následujících grafech a tabulkách jsou vždy převedeny zpět na hodnotu funkce f.

5.1 Přehled používaných funkcí

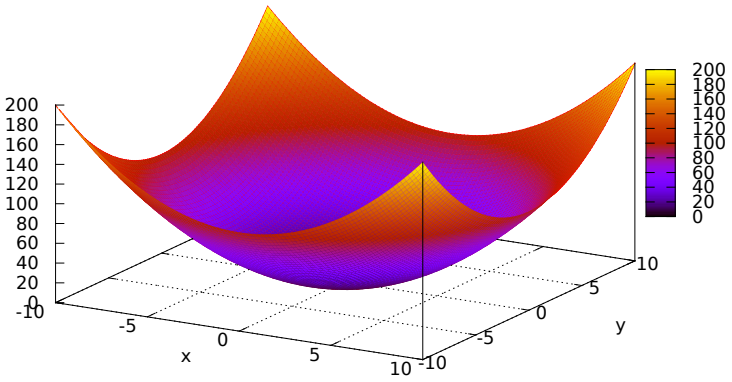
Při testování nových optimalizačních algoritmů se obvykle vychází z již dříve provedených experimentů. Metoda LEM a z ní odvozené modifikace byly ověřovány na problémech z oblasti numerické optimalizace. Rozhodl jsem se pro sadu šesti funkcí, každá z nich má vlastnosti, jimiž se odlišuje od ostatních.

5.1.1 Sférická funkce

Funkce má jedno minimum o hodnotě 0 pro $\forall x_i = 0$.

$$f_1(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i^2 \qquad -600 \leq x_i \leq 600 \tag{5.1}$$

Jedná se o velmi jednoduchou funkci, selhává-li na ní optimalizační algoritmus, pak lze předpokládat, že nebude úspěšný ani při optimalizaci jiných funkcí.



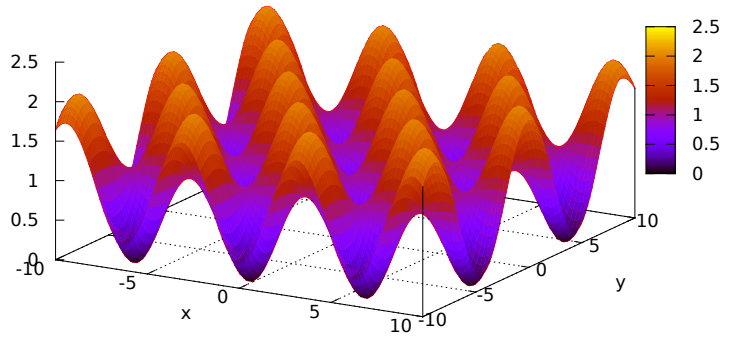
Obrázek 5.1: Sférická funkce dvou proměnných

5.1.2 Griewangkova funkce

Tato funkce má řadu lokálních minim a jedno minimum globální pro $\forall x_i = 0$.

$$f_2(x_1, \ldots, x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \qquad -600 \leq x_i \leq 600 \tag{5.2}$$

Lokální minima této funkce jsou pravidelně rozložena po celé ploše. Hodnoty minim se od sebe výrazně neodlišují, což činí nalezení globálního minima obtížnější. Nevýhodou této funkce je fakt, že s počtem dimenzí složitost optimalizace klesá, čímž se tato funkce odlišuje od všech ostatních.



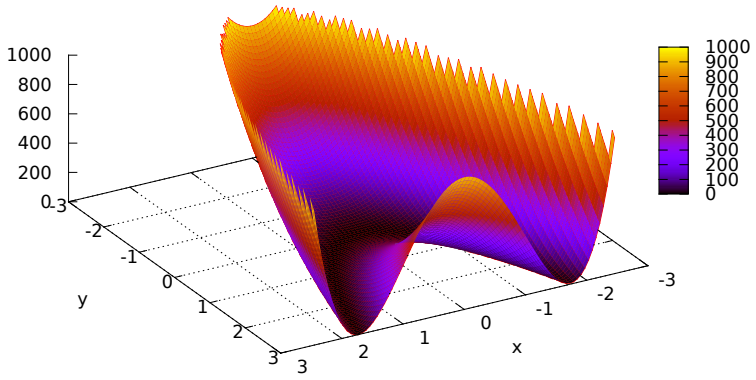
Obrázek 5.2: Griewangkova funkce dvou proměnných

5.1.3 Rosenbrockova funkce

Funkce má jedno minimum s hodnotou 0 pro $\forall x_i = 1$.

$$f_3(x_1, \ldots, x_n) = \sum_{i=1}^{n-1} (100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \qquad -10 \leq x_i \leq 10 \qquad (5.3)$$

Rosenbrockova funkce je specifická vzájemnou závislostí jednotlivých proměnných. Globální minimum se nalézá uvnitř úzké plošší oblasti, mimo niž hodnoty prudce vzrůstají. Tato vlastnost činí jeho nalezení obtížnější.



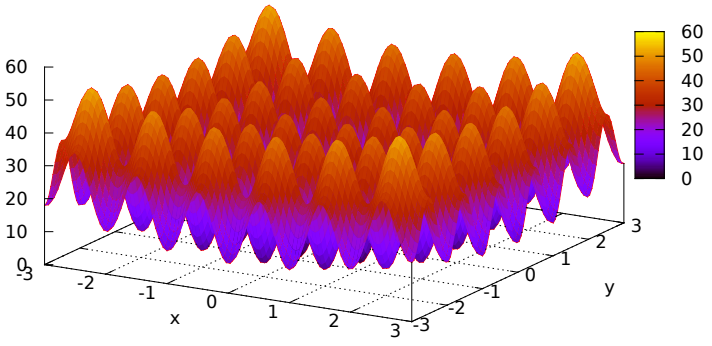
Obrázek 5.3: Rosenbrockova funkce dvou proměnných

5.1.4 Rastriginova funkce

Další funkce s řadou lokálních minim v okolí globálního minima, které se nachází pro $\forall x_i = 0$ a má hodnotu 0.

$$f_4(x_1, \ldots, x_n) = 10 \cdot 10 + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \qquad -5{,}12 \leq x_i \leq 5{,}12 \qquad (5.4)$$

Rastriginova funkce má díky kosinu hodnotově výraznější lokální minima, než např. Griewangkova funkce.



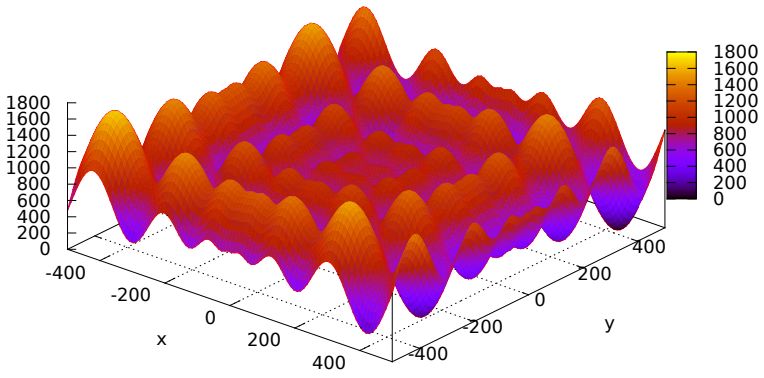
Obrázek 5.4: Rastriginova funkce dvou proměnných

5.1.5 Schwefelova funkce

Funkce s jedním globálním minimem pro $\forall x_i \doteq 420{,}968746$.

$$f_5(x_1, \ldots, x_n) = 418{,}9828873n - \sum_{i=1}^{n} x_i \sin(\sqrt{x_i}) \qquad -500 \leq x_i \leq 500 \tag{5.5}$$

V definovaném prostoru se minima nachází v krajních oblastech.

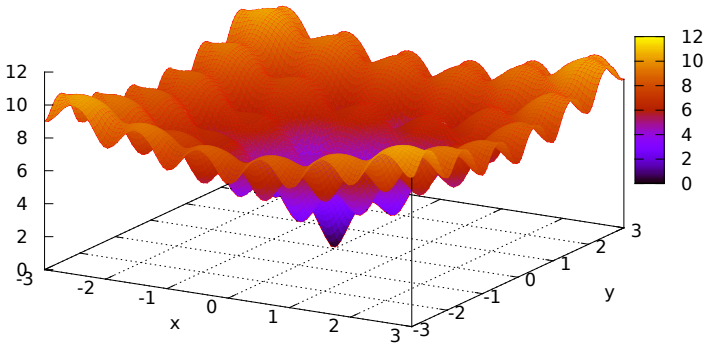


Obrázek 5.5: Schwefelova funkce dvou proměnných

5.1.6 Ackleyho funkce

Tato funkce má mnoho lokálních minim v okolí minima globálního, jež má hodnotu 0 pro $\forall x_i = 0$.

$$f_6(x_1, \ldots, x_n) = 20 + e - 20e^{-0{,}2 \cdot \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sum_{i=1}^{n} \cos(2\pi x_i)}{n}} \qquad -32{,}768 \leq x_i \leq 32{,}768 \tag{5.6}$$



Obrázek 5.6: Ackleyho funkce dvou proměnných

5.2 Velikost skupiny

Počet jedinců tvořících skupinu, na níž je trénován klasifikátor, hraje důležitou roli. V klasifikačních úlohách často větší trénovací sada vede k lepším výsledkům klasifikátoru na testovací sadě a v reálném nasazení. Avšak tvorba populace z klasifikátoru představuje zcela jiný problém. S velikostí trénovací sady zcela určitě bude růst obecnost pravidel, která budou popisovat nejlepší jedince.

Velikost skupiny v dřívějších variantách metody prakticky nebyla diskutována. Obvykle bylo zmíněno, že skupina obsahuje 30% populace, nicméně tato hodnota nebyla rozumně zdůvodněna.

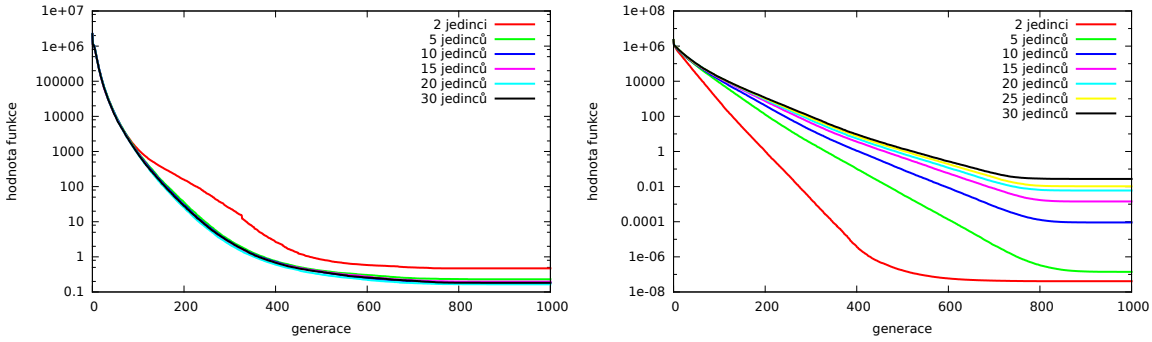
Cílem tohoto experimentu je zjistit vhodnou velikost skupin použitých pro trénování klasifikátoru. Velikost skupiny se pohybuje od pouhých 2 jedinců až po 30 jedinců. Do grafů (obrázky 5.7 - 5.12) jsou vyneseny průměrné nejlepší hodnoty pro každou generaci.

Parametry experimentu jsou následující:

Dimenze testovacích funkcí	30
Počet generací	1000
Počet opakovaných běhů	100
Velikost populace	100 jedinců
Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	2

5.2.1 Výsledky

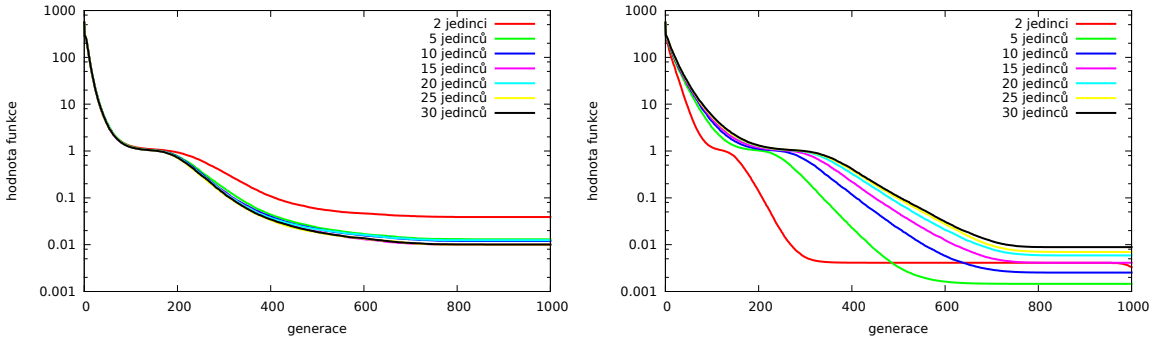
Už na optimalizaci jednoduché **sférické** funkce se ukázalo být vhodné zkoumat velikost skupin. Pro LEM s SVM klasifikátorem byla dosahovaná hodnota lepší s menším počtem jedinců na skupinu. Při větších velikostech skupiny populace konvergovala pomaleji a dosahovala výrazně horších hodnot. Situace u LEM s AdaBoost však byla diametrálně odlišná, nejmenší skupina si vedla nejhůře, jinak počet jedinců neměl vliv ani na rychlost ani na kvalitu dosahovaného řešení. To je však velmi vzdáleno od hodnot dosahovaných LEM(SVM).



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.7: Porovnání dosahovaných hodnot sférické funkce pro různé velikosti skupiny

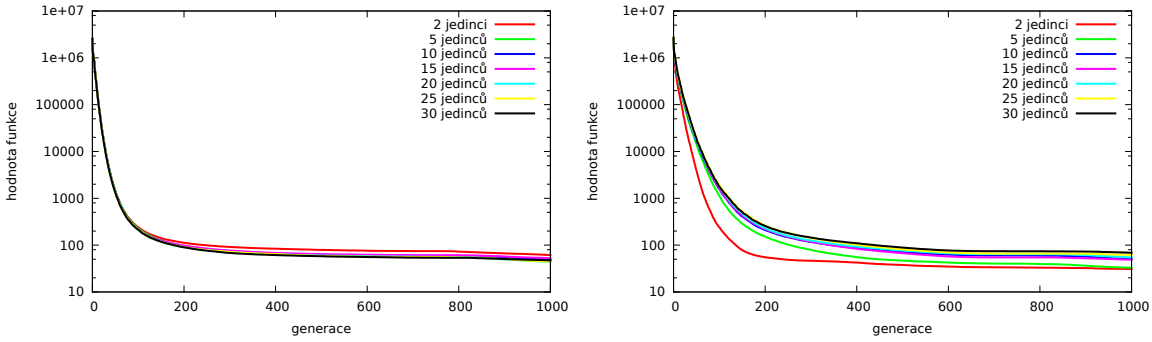
U **Griewangkovy** funkce se metoda LEM(AdaBoost) chová podobně jako v předchozím případě. Nejmenší skupina vykazuje nejhorší výsledky, ostatní velikosti jsou srovnatelné. Zajímavější stav je u LEM(SVM), kdy skupina o dvou jedincích má nejrychlejší směřování k minimu. Po třech stech generacích dosahuje hodnoty lepší než kterákoli varianta LEM(AdaBoost), nicméně v tu dobu již konvergence ustává a následně je v pěti sté generaci překonána variantou s pěti a o něco později i variantou s deseti jedinci na skupinu. Větší velikosti skupin zaostávají. Celkově se optimu nejvíce přiblížila LEM(SVM) se skupinami o pěti jedincích.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.8: Porovnání dosahovaných hodnot Griewangkovy funkce pro různé velikosti skupiny

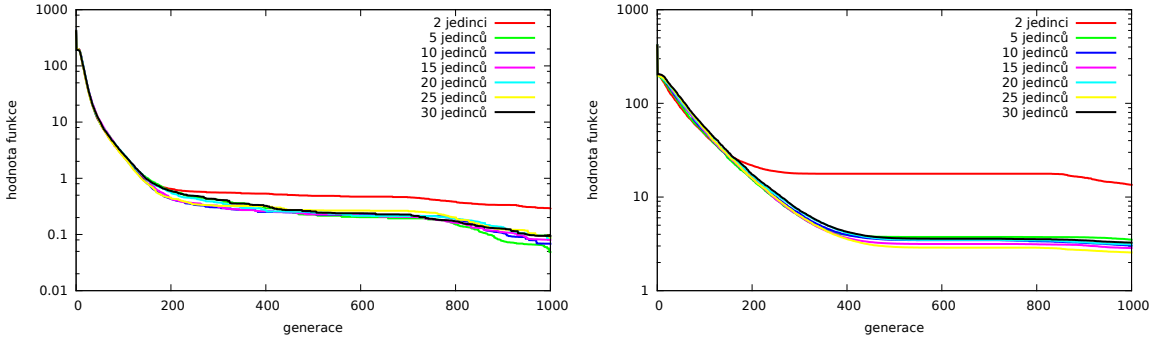
Při hledání optima **Rosenbrockovy** funkce si obě metody nevedly příliš dobře. Konvergence se výrazně zpomaluje po dvou sté generaci. Rozdíly mezi velikostmi skupin jsou opět patrné jen u LEM s SVM klasifikátorem, kde dosahují dvou a pěti členné skupiny stejné hodnoty, menší z nich přibližně dvakrát rychleji.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.9: Porovnání dosahovaných hodnot Rosenbrockovy funkce pro různé velikosti skupiny

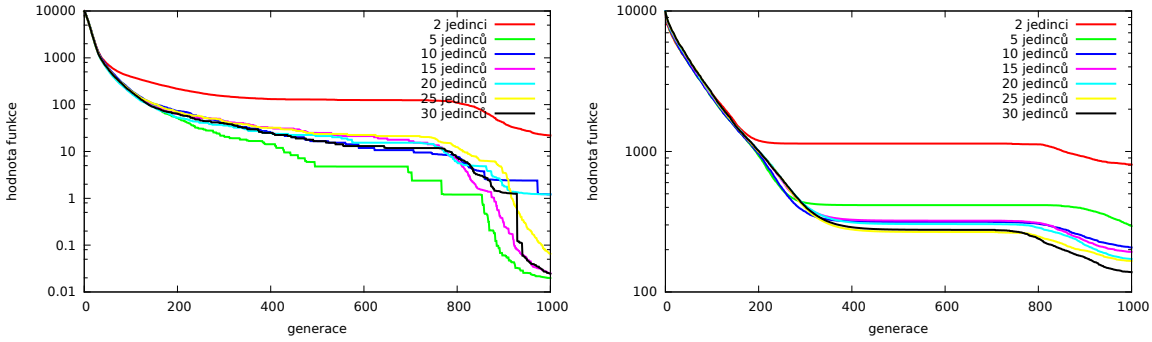
Na rozdíl od předchozích případů si při optimalizaci **Rastriginovy** funkce vedla nejhůře varianta se skupinou o dvou jedincích i pro LEM(SVM). Větší velikosti skupin dosahují obdobných výsledků. LEM(AdaBoost) zcela překonává LEM(SVM), už po dvou stech generacích je dosažená hodnota výrazně lepší, než které LEM(SVM) dosáhne po tisíci generacích.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.10: Porovnání dosahovaných hodnot Rastriginovy funkce pro různé velikosti skupiny

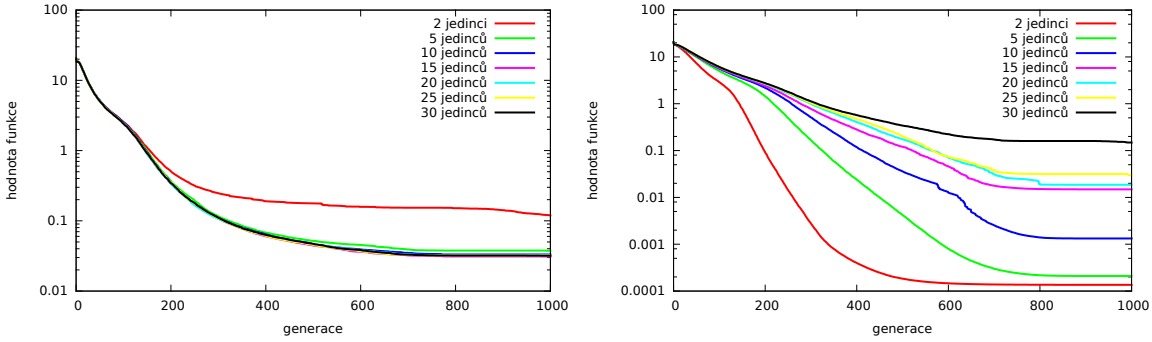
I u hledání minima **Schwefelovy** funkce platí, že nejmenší skupina dosahuje nejméně kvalitních výsledků. U LEM(SVM) se nalézané hodnoty zlepšují s velikostí skupiny, avšak opět jsou velmi daleko od optima. Zajímavější situace je u LEM s klasifikátorem AdaBoost, kde si nejlépe vede varianta s pěti jedinci ve skupině, takřka stejných hodnot ale dosáhne i největší skupina o třiceti jedincích. Nejhůře se daří při použití dvou, deseti či dvaceti členných skupin.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.11: Porovnání dosahovaných hodnot Schwefelovy funkce pro různé velikosti skupiny

Při optimalizaci **Ackleyho** funkce se dostavil podobný průběh jako u Griewangkovy nebo sférické funkce. Pro LEM(SVM) je nejlepší nejmenší velikost skupin, pak jsou dosahované hodnoty horší s počtem jedinců. U LEM(AdaBoost) není rozdíl ve výkonnosti při skupině o pěti a více jedincích, nejmenší skupina si vede nejhůře.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.12: Porovnání dosahovaných hodnot Ackleyho funkce pro různé velikosti skupiny

	počet jedinců ve skupině						
	2	5	10	15	20	25	30
LEM(AdaBoost)	0,4	0,84	2,85	5,46	8,72	12,6	18,1
LEM(SVM)	0,38	0,44	1,01	2,9	4,23	8,3	15,36

Tabulka 5.1: Průměrná doba běhu v sekundách pro různé velikosti skupin

Experimenty ukázaly, že je důležité uvažovat nad počtem jedinců ve skupině. Velikostí trénovacích skupin totiž ovlivňujeme i dobu optimalizační běhu (tabulka 5.1).

Pro **LEM s AdaBoost** klasifikátorem byl optimalizační průběh shodný takřka pro všechny velikosti kromě té nejmenší, proto je s ohledem na čas textbfnejvhodnější používat skupinu **o pěti jedincích**. U **LEM s SVM** je situace jiná, velikost přímo ovlivnila kvalitu

řešení. Nejlepší výsledky byly při menších velikostech. Skupiny **o dvou jedincích** byly v konvergenci rychlejší, ale pětičlenná varianta se jí velmi blížila a v případě Rastriginovy a Schwefelovy funkce ji předstihla.

5.3 Více skupin

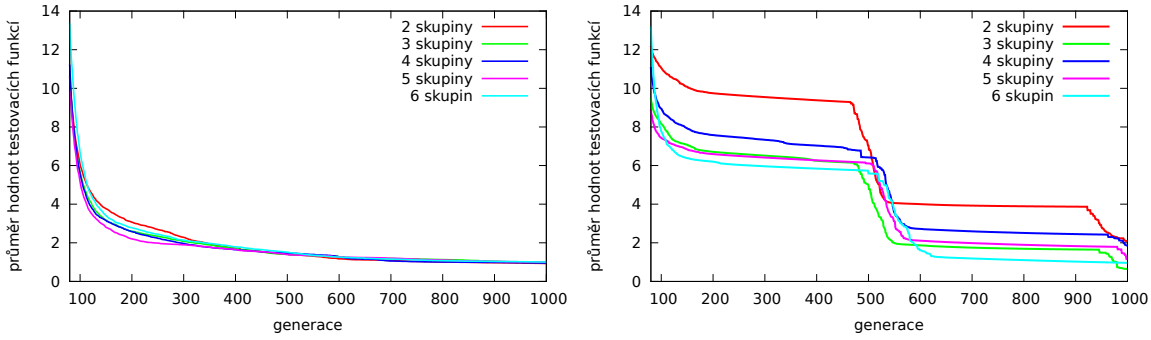
Technika s použitím více skupin se neobjevila v žádné předchozí verzi metody LEM. Cílem experimentu je zjistit, zda vytváření jedinců podle většího počtu pravidel přinese rozdíl v podobě změny rychlosti konvergence či ve větší kvalitě dosaženého řešení.

Pro experiment byly použity následující parametry:

Dimenze testovacích funkcí	$10, 30$
Počet generací	1000
Počet opakovaných běhů	100
Velikost populace	100 jedinců
Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	$2, 3, 4, 5, 6$
Velikost skupiny	5 jedinců

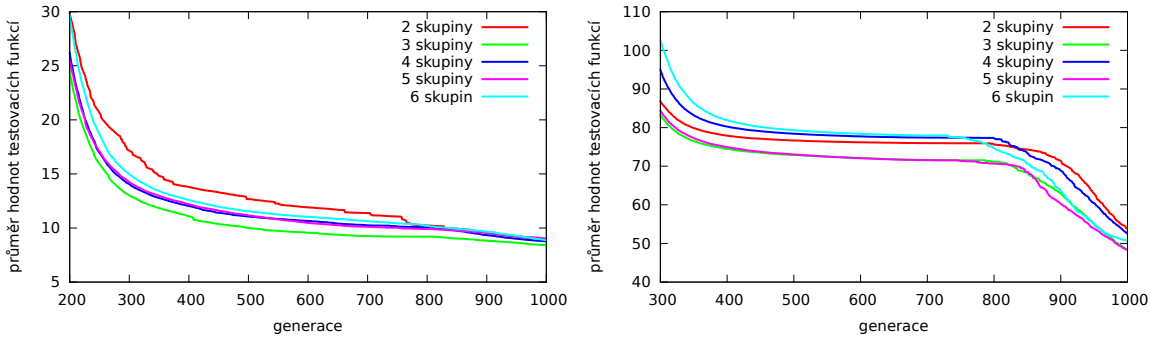
Pro zjištění celkové výkonnosti určitého počtu skupin se používal průměr dosažených hodnot přes všechny testovací funkce. Je zřejmé, že některé funkce budou mít větší vliv na průměrnou hodnotu než ostatní, ale jelikož se porovnává změna pouze jednoho parametru metody, nemělo by to významně ovlivnit výsledek.

Tyto průměrné hodnoty pro každou generaci jsou vyneseny do příslušných grafů v obrázcích 5.13 pro deset dimenzí a 5.14 pro třicet dimenzí.



(a) LEM(AdaBoost) (b) LEM(SVM)

Obrázek 5.13: Porovnání vlivu počtu skupin na konvergenci pro deset dimenzí



(a) LEM(AdaBoost)

(b) LEM(SVM)

Obrázek 5.14: Porovnání vlivu počtu skupin na konvergenci pro třicet dimenzí

	počet skupin				
	2	3	4	5	6
LEM(AdaBoost)	0,27	0,68	1,05	1,72	2,74
LEM(SVM)	0,17	0,23	0,31	0,41	0,55

Tabulka 5.2: Průměrná doba běhu v sekundách pro různé počty skupin, dimenze $N = 10$

Pro deset dimenzí (obrázek 5.13) zůstala metoda LEM(AdaBoost), obdobně jako v experimentu s velikostmi skupin, **neovlivněna** zkoumaným parametrem. Určité rozdíly se vyskytují do třísté generace, kde si vede nejhůře varianta se dvěma skupinami. Po té ale rozdíly zcela mizí. U LEM(SVM) jsou rozdíly mezi jednotlivými variantami znatelnější, při použití dvou skupin je ztráta největší, velmi dobře si vede už tří skupinová varianta. Pro třicet dimenzí (obrázek 5.14) jsou rozdíly mezi skupinami znatelnější i pro LEM s klasifikátorem AdaBoost, nicméně při konci evolučního procesu jsou vzájemné odchylky minimální.

S přihlédnutím k době běhu (tabulka 5.2) je vhodné používat **dvě skupiny pro LEM s AdaBoost** klasifikátorem a **tři skupiny pro LEM s SVM** klasifikátorem.

5.4 Vliv evoluční části

Cílem experimentu je určit, jak ovlivňují výkonnost evoluční operátory křížení a mutace, zda je vhodné či přímo nutné evoluční fázi používat. Křížení a mutace totiž může vhodně kombinovat jedince vytvářené dle klasifikátoru a připravit tak vhodné jedince pro trénování klasifikátoru v další generaci.

Parametry experimentu:

Dimenze testovacích funkcí	10
Počet generací	1000
Počet opakovaných běhů	100
Velikost populace	100 jedinců
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	2
Velikost skupin	5 jedinců

Porovnávala se dosažená hodnota po 200. a 1000. generaci při variantě bez evoluční části (tj. 90% jedinců generovaných dle klasifikátoru) a ve spolupráci s evoluční částí (tj. 70% jedinců generovaných dle klasifikátoru, 20% pomocí křížení a mutace). Výsledky jsou uvedeny v tabulce 5.3.

Stav po 200. generaci	LEM(AdaBoost)		LEM(SVM)	
funkce	s evolucí	bez evoluce	s evolucí	bez evoluce
sférická	$2{,}7E{-}2$	$4{,}7E{+}1$	$4{,}2E{-}9$	$7{,}2E{-}8$
Griewangkova	$7{,}3E{-}2$	$8{,}1E{-}1$	$9{,}6E{-}2$	$2{,}3E{-}1$
Rosenbrockova	$1{,}8E{+}1$	$4{,}6E{+}1$	5,42	9,25
Rastriginova	$1{,}2E{-}2$	1,43	1,54	6,03
Schwefelova	1,19	3,25	$5{,}3E{+}1$	$1{,}8E{+}2$
Ackleyho	$1{,}8E{-}2$	$9{,}3E{-}1$	$4{,}6E{-}5$	$3{,}1E{-}4$

Stav po 1000. generaci	LEM(AdaBoost)		LEM(SVM)	
funkce	s evolucí	bez evoluce	s evolucí	bez evoluce
sférická	$7{,}3E{-}5$	1,23	0,0	$1{,}2E{-}8$
Griewangkova	$8{,}6E{-}3$	$2{,}3E{-}1$	$4{,}8E{-}2$	$8{,}9E{-}2$
Rosenbrockova	5,77	8,53	$9{,}3E{-}1$	1,1
Rastriginova	$3{,}1E{-}6$	$3{,}6E{-}1$	$7{,}9E{-}1$	2,48
Schwefelova	$3{,}1E{-}6$	$6{,}7E{-}2$	$1{,}3E{+}1$	$9{,}2E{+}1$
Ackleyho	$8{,}3E{-}4$	$3{,}5E{-}1$	$3{,}8E{-}6$	$1{,}4E{-}4$

Tabulka 5.3: Srovnání vlivu evoluční části

Výsledky experimentu dávají jednoznačnou odpověď, **zapojení evolučních operátorů je více než vhodné**. Ve všech případech byla varianta s evoluční částí lepší. Rozdíl je patrný už po dvě stě generacích. Po tisíci generacích bylo zlepšení varianty bez evoluce i o několik řádů horší než u druhé varianty.

5.5 Srovnání LEM s původní LEM3

Cílem experimentu bylo zjistit, zda provedené modifikace metody přinesly pokrok ve smyslu dosahování lepších hodnot.

Rovněž jsem považoval za vhodné ověřit, zda metoda diskretizace spojité proměnné použitá v LEM3 má takový význam, aby byla implementována i v LEM s klasifikátory AdaBoost a SVM. Za tímto účelem jsem provedl porovnání všech tří metod na testovacích funkcích, které byly posunuty o náhodně vygenerovaný vektor.

Spustitelná implementace metody LEM3 byla získána z http://www.mli.gmu.edu/download/lem3.zip.

Parametry LEM(AdaBoost) a LEM(SVM):

Dimenze testovacích funkcí	10
Počet generací	600
Počet opakovaných běhů	100
Velikost populace	100 jedinců
Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	2
Velikost skupiny	5 jedinců

U LEM3 byly ponechány výchozí doporučené hodnoty parametrů.

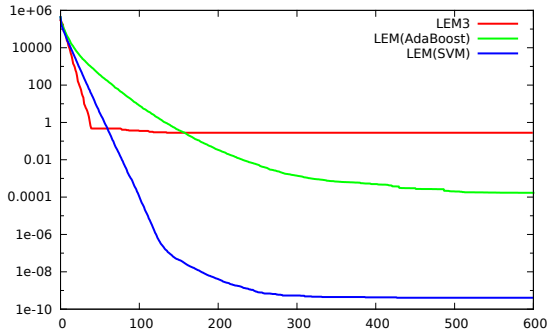
V případě neposunutých funkcí LEM3 našla přesnou pozici globálního minima (až na Schwefelovu funkci) do dvou set generací, často i do deseti generací. Čím menší byl definiční obor, tím bylo dosaženo rychleji. LEM s AdaBoost či SVM klasifikátorem v tomto srovnání zcela neobstojí (dosažené hodnoty v tabulce 5.3).

Pohled na vývoj dosahovaných hodnot (obrázek 5.15) při hledání minima posunutých variant testovacích funkcí jednoznačné ukazuje, že LEM3 v konvergenci výrazně zpomalí už po 100. generaci. Z hodnot získaných po 600. generaci (tabulka 5.4) je vidět, že **LEM s AdaBoost** klasifikátorem **dosahuje lepších hodnot než LEM3** (až na Rosenbrockovu funkci). Metoda LEM s SVM dosáhla horšího řešení pouze u Rastriginovy a Schwefelovy funkce.

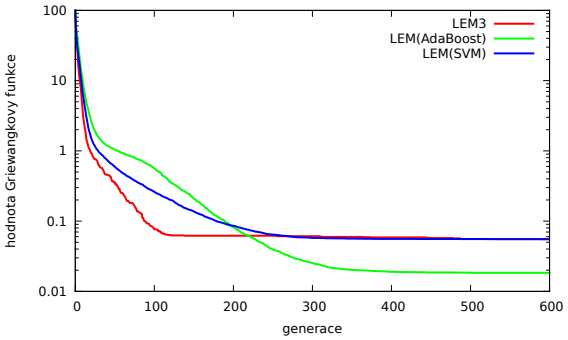
LEM3 díky diskretizační metodě nalezla nejbližší celočíselné řešení relativně rychle, ale už nebyla schopna je zlepšit. Proto jsem se rozhodl metodu diskretizace spojité proměnné nepoužívat.

funkce	LEM3	LEM(AdaBoost)	LEM(SVM)
sférická	$2{,}8E{-}1$	$1{,}7E{-}4$	$4{,}1E{-}10$
Griewangkova	$5{,}6E{-}2$	$1{,}8E{-}2$	$5{,}5E{-}2$
Rosenbrockova	8,1	8,2	2,79
Rastriginova	$1{,}97E{-}1$	$3E{-}2$	1,44
Schwefelova	$1{,}4E{+}1$	$1{,}3E{-}5$	$2{,}1E{+}1$
Ackleyho	$2{,}5E{-}1$	$1{,}7E{-}3$	$1{,}1E{-}5$

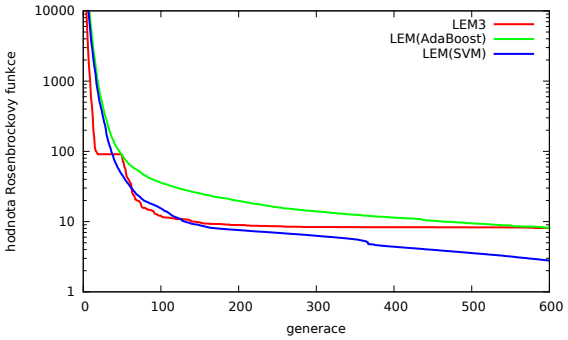
Tabulka 5.4: Dosažené hodnoty po 600. generaci, funkce byly posunuty o náhodně vygenerovaný vektor



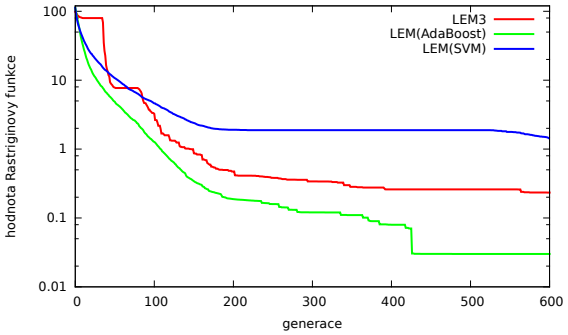
(a) sférická funkce



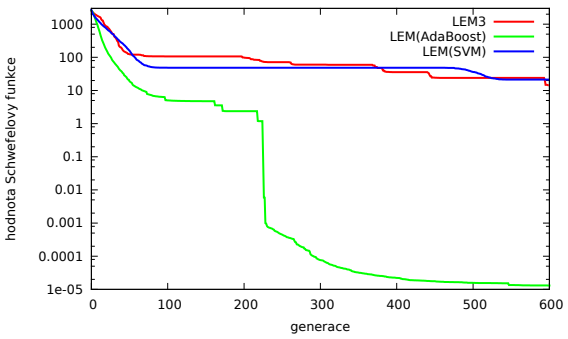
(b) Griewangkova funkce



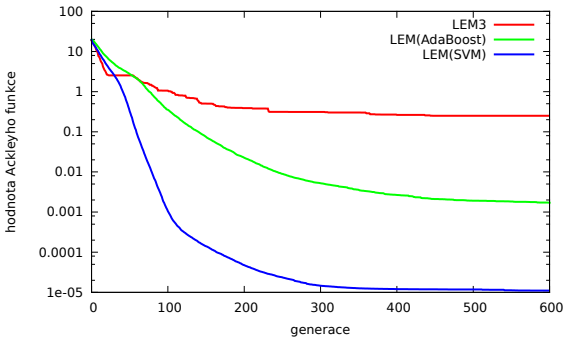
(c) Rosenbrockova funkce



(d) Rastriginova funkce



(e) Schwefelova funkce



(f) Ackleyho funkce

Obrázek 5.15: Srovnání LEM(AdaBoost) a LEM(SVM) s LEM3 na posunutých funkcích

5.6 Porovnání metod LEM s EDA algoritmem

Cílem posledního z experimentů ve statickém prostředí bylo využít získaných znalostí z předchozích experimentů a aplikovat je při srovnání s EDA algoritmem MBOA (Mixed Bayesian Optimization Algorithm) [17]. Zároveň se ověřuje chování LEM se střídáním klasifikátorů (tj. LEM(AdaBoost+SVM).

EDA (Estimation of Distribution Algorithm) algoritmy jsou postaveny na pravděpodobnostní teorii. Nové jedince generují dle pravděpodobnostního modelu, který je vytvořen z nejvýhodnější části populace. Tím je tento přístup podobný evolučnímu modelu s učením.

Algoritmus MBOA používá množinu rozhodovacích stromů k vytvoření pravděpodobnostního modelu.

Zdrojové kódy implementace algoritmu MBOA byly získány z `http://jiri.ocenasek.com/code/MBOA.zip`.

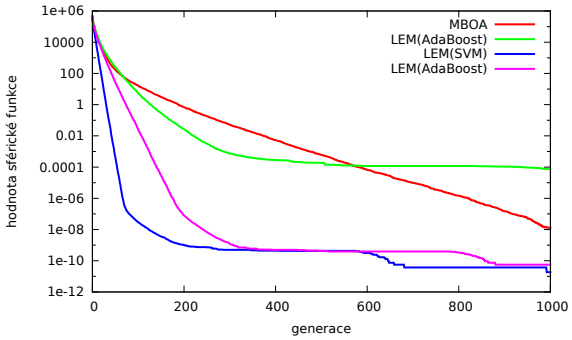
Parametry experimentu:

Dimenze testovacích funkcí $[N]$	$10, 30, 50$
Počet generací	1000 pro $N = 10$, 3000 pro ostatní
Počet opakovaných běhů	100
Velikost populace	100 jedinců
Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	3 pro LEM(SVM), 2 pro ostatní
Velikost skupiny	2 pro LEM(SVM), 5 pro ostatní

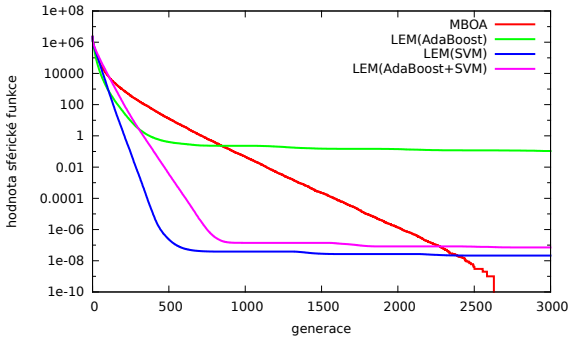
5.6.1 Hledání minima sférické funkce

Přestože je definice sférické funkce velmi jednoduchá a dalo by se tedy očekávat, že pro všechny zkoumané algoritmy bude nalezení minima snadné, není tomu tak. LEM s klasifikátorem AdaBoost zaostává oproti ostatním.

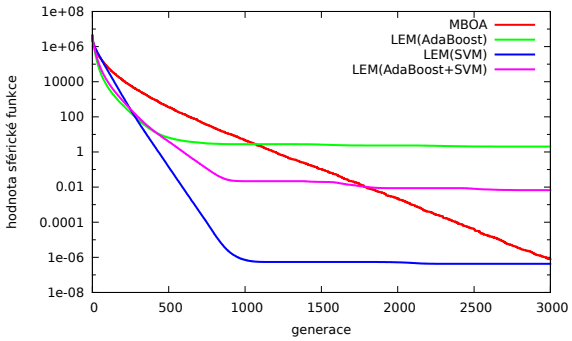
Po skončení optimalizačního procesu jsou sice hodnoty dosažené MBOA a LEM s SVM velmi blízké, ale pohled na grafy závislosti hodnoty funkce na generaci (obrázek 5.16) odhaluje, že LEM s SVM klasifikátorem je několikanásobně rychlejší. Obě metody LEM mají společné, že po rychlém dosažení kvalitní hodnoty se už příliš nedokáží zlepšit. Naproti tomu MBOA postupuje k optimu pomaleji a je pravděpodobné, že by při větším počtu generací předstihla LEM s SVM ve všech případech. Kombinovaná varianta LEM se pro 10 a 30 dimenzí výkonnostně blíží LEM s SVM, pro 50 je přibližně uprostřed mezi LEM(AdaBoost) a LEM(SVM).



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	$1{,}3E{-}8$	**0,0**	$7{,}84E{-}7$
LEM(AdaBoost)	$7{,}38E{-}5$	$1{,}07E{-}1$	2,04
LEM(SVM)	**1,86E−11**	$2{,}15E{-}8$	**4,29E−7**
LEM(AdaBoost+SVM)	$5{,}59E{-}11$	$1{,}29E{-}7$	$6{,}68E{-}3$

(d) dosažené hodnoty

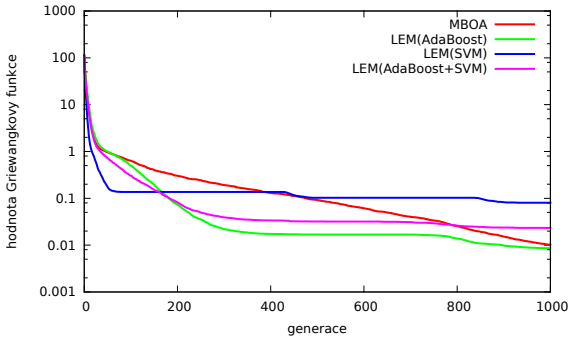
Obrázek 5.16: Vývoj hodnot sférické funkce

5.6.2 Hledání minima Griewangkovy funkce

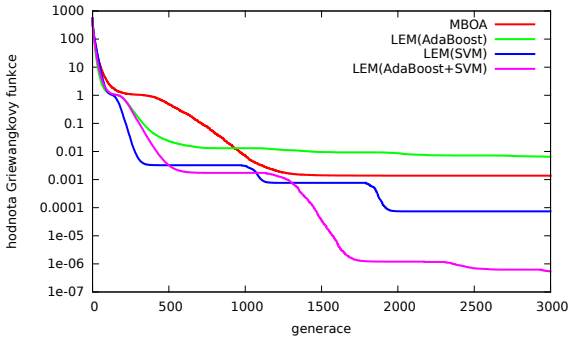
V tomto případě je výrazný rozdíl v chování algoritmů v závislosti na počtu dimenzí. Pro deset dimenzí je průběh optimalizace podobný jako u sférické funkce, s tím rozdílem, že LEM(AdaBoost) překonává LEM(SVM). U třiceti a padesáti dimenziální varianty se situace obrací a LEM(SVM) je opět lepší. Také je zajímavé sledovat její „schodovitý“ průběh (obrázek 5.17 (b) a (c)), díky němuž není překonána algoritmem MBOA.

Rozdíl v rychlosti konvergence není sice tak značný jako u předešlé funkce, ale pro třicet a padesát dimenzí stačí LEM s SVM poloviční počet generací oproti MBOA.

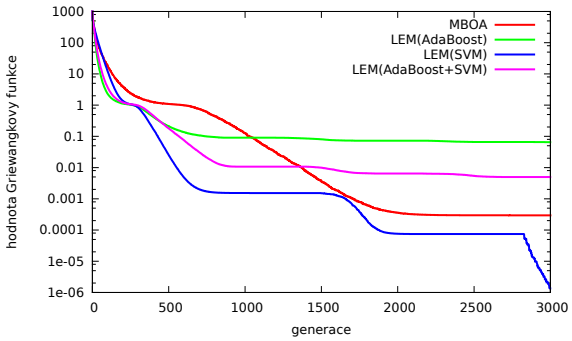
Kombinovaná varianta LEM(AdaBoost+SVM) dosahuje nejlepších hodnot při třiceti dimenzích, v ostatních variantách je opět výkonnostně mezi LEM(AdaBoost) a LEM(SVM).



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	$1{,}02E{-}2$	$1{,}38E{-}3$	$2{,}96E{-}4$
LEM(AdaBoost)	**8,59E−3**	$6{,}51E{-}3$	$6{,}52E{-}2$
LEM(SVM)	$8{,}05E{-}2$	$7{,}4E{-}5$	**1,35E−6**
LEM(AdaBoost+SVM)	$2{,}32E{-}2$	**5,38E−7**	$4{,}99E{-}3$

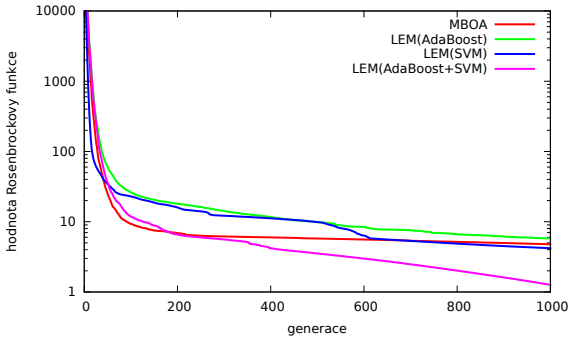
(d) dosažené hodnoty

Obrázek 5.17: Vývoj hodnot Griewangkovy funkce

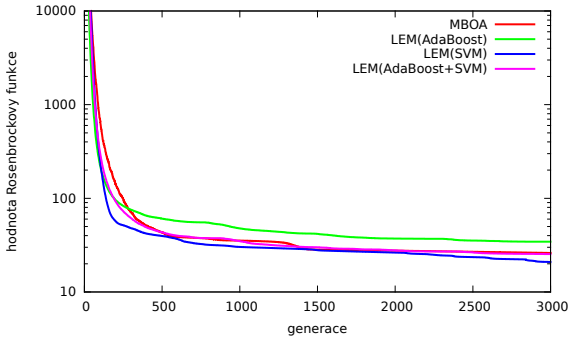
5.6.3 Hledání minima Rosenbrockovy funkce

Rosenbrockova funkce byla pro porovnávané metody stejně obtížná. K minimu se neblíží žádný. Lze říct, že konvergují k hodnotě $\forall x_i = 0$, kdy má Rosenbrockova funkce hodnotu rovnu počtu dimenzí. Důvodem k selhání optimalizace je pravděpodobně závislost mezi proměnnými.

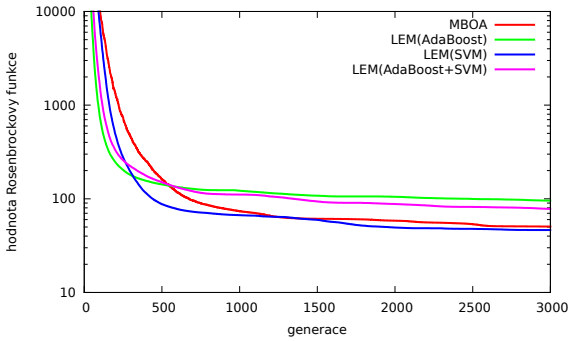
Pro deset dimenzí je nejrychlejší algoritmem LEM(AdaBoost+SVM). Mezi ostatními algoritmy jsou rozdíly přibližně do 600. generace, kdy se dosahované hodnoty srovnají. Ve všech případech LEM s SVM mírně překoná MBOA a LEM s klasifikátorem AdaBoost je vždy nejhorší ze všech variant. Kombinovaná varianta dosáhne výrazně lepší hodnoty při 10 dimenzích, v ostatních případech je horší než LEM(SVM).



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	4,78	$2{,}61E{+}1$	$5{,}05E{+}1$
LEM(AdaBoost)	5,78	$3{,}44E{+}1$	$9{,}56E{+}1$
LEM(SVM)	4,21	**2,09E+1**	**4,64E+1**
LEM(AdaBoost+SVM)	**1,26**	$2{,}54E{+}1$	$7{,}78E{+}1$

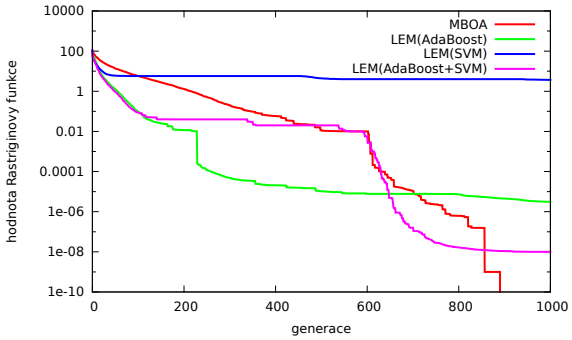
(d) dosažené hodnoty

Obrázek 5.18: Vývoj hodnot Rosenbrockovy funkce

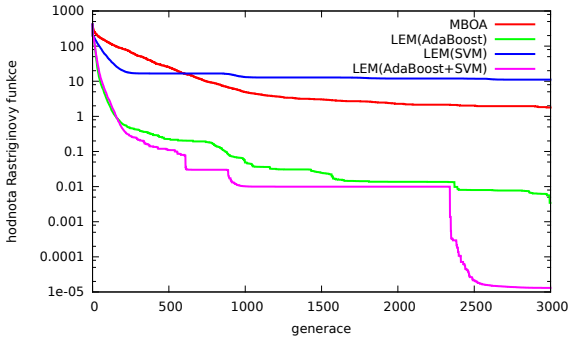
5.6.4 Hledání minima Rastriginovy funkce

Tentokráte je LEM(AdaBoost+SVM) ve všech případech lepší než LEM(AdaBoost) a výrazně lepší než LEM(SVM). Při deseti dimenzích dosahuje obdobných hodnot jako MBOA přibližně do 800. generace, kdy MBOA dosáhne globálního minima. Při třiceti dimenzích se situace mění, žádná z metod nedosahuje minima, ale obě LEM využívající AdaBoost jsou po dvě stě generacích na stejné hodnotě jako MBOA po tisíci.

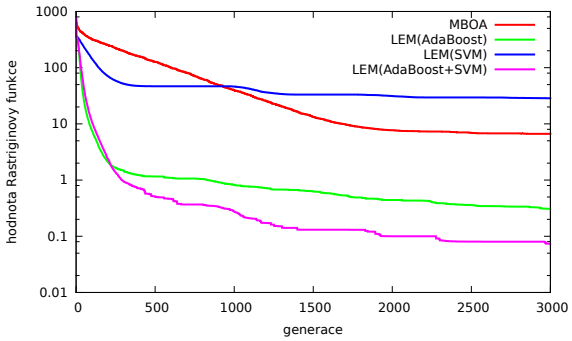
U padesáti dimenzionální varianty je rozdíl ještě větší. LEM(AdaBoost+SVM) po méně než sto generacích dosahuje obdobné hodnoty jako LEM(SVM) po čtyř stech generacích a MBOA po tisíci generacích. Po pěti sté generaci se už nalezená hodnota LEM(SVM) nezlepší, MBOA ustane ve směřování k minimu při dosažení 2000. generace, LEM(AdaBoost) této hodnoty dosáhne desetkrát rychleji.



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	**0,0**	1,79	6,65
LEM(AdaBoost)	$3{,}09E{-}6$	$3{,}43E{-}3$	$3{,}07E{-}1$
LEM(SVM)	3,68	$1{,}11E{+}1$	$2{,}85E{+}1$
LEM(AdaBoost+SVM)	$9{,}98E{-}9$	**1,31E−5**	**7,34E−2**

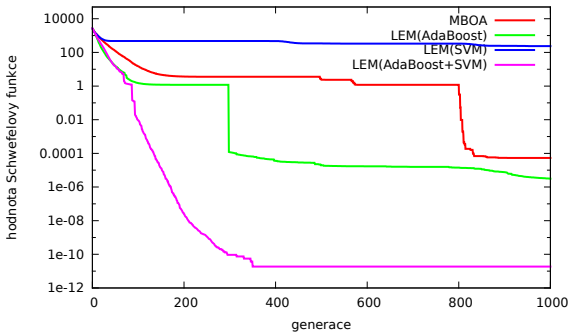
(d) dosažené hodnoty

Obrázek 5.19: Vývoj hodnot Rastriginovy funkce

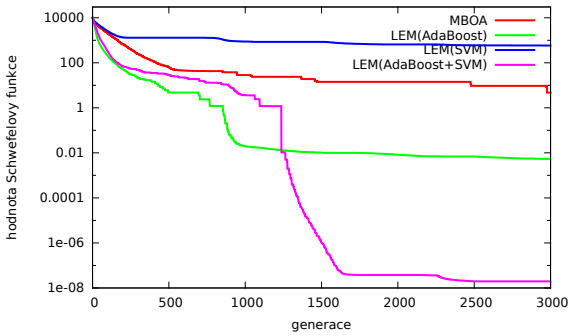
5.6.5 Hledání minima Schwefelovy funkce

Hledání minima se ukázalo být doménou LEM(AdaBoost+SVM), velmi dobře si vede i LEM(AdaBoost). Kombinovaná varianta se nezávisle na počtu dimenzí se k optimu vždy blíží, MBOA i LEM(SVM) trpí problémem uváznutí v lokálním extrému. U Schwefelovy funkce je to značný problém, jelikož hodnota lokálního minima je o několik řádů horší než hodnoty v okolí globálního minima. Tím je pak samozřejmě ovlivněna průměrná dosažená hodnota.

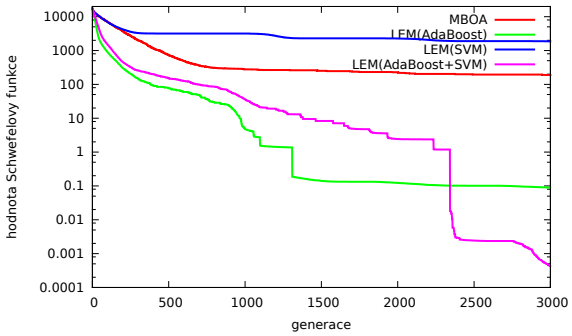
Celkově LEM(SVM) zcela selhává, část nalezeného řešení zůstane v lokálním extrému. MBOA je srovnatelná s LEM(AdaBoost) pouze při deseti dimenzích (kdy je ovšem výrazně pomalejší), s počtem dimenzí se rozdíl mezi nimi výrazně zvětšuje.



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	$5{,}34E{,}5$	$4{,}74$	$1{,}9E{+}2$
LEM(AdaBoost)	$3{,}14E{-}6$	$5{,}38E{-}3$	$8{,}99E{-}2$
LEM(SVM)	$2{,}37E{+}2$	$5{,}82E{+}2$	$1{,}86E{+}3$
LEM(AdaBoost+SVM)	$\mathbf{1{,}86E{-}11}$	$\mathbf{1{,}97E{-}8}$	$\mathbf{4{,}32E{-}4}$

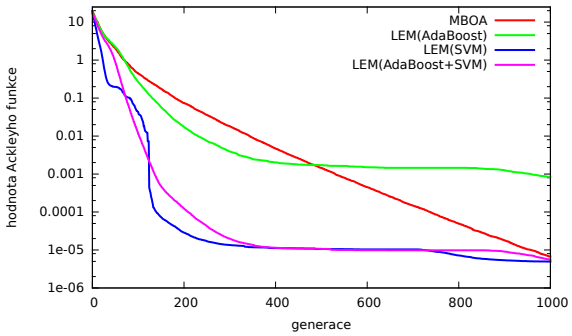
(d) dosažené hodnoty

Obrázek 5.20: Vývoj hodnot Schwefelovy funkce

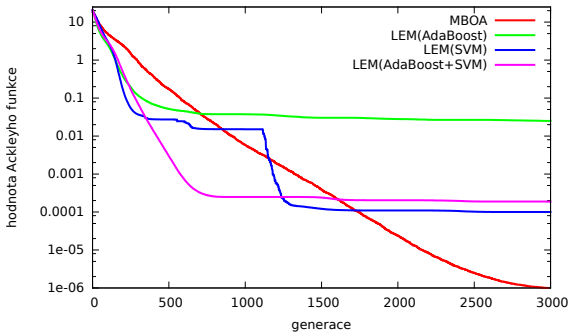
5.6.6 Hledání minima Ackleyho funkce

V tomto případě si vedla lépe metoda LEM(SVM) než LEM(AdaBoost). Markantní rozdíl byl už při deseti dimenzích, kdy po dvou stech generacích dosahuje LEM(SVM) obdobné hodnoty jako MBOA po osmi stech a výrazně lepší hodnoty než, které LEM(AdaBoost) dosáhne na konci. Pro vícedimenzionální varianty je výkonnost LEM(AdaBoost) a LEM(SVM) podobná během prvních tisíc generací, ale obě zaostávají za LEM(AdaBoost+SVM). Ve druhé polovině optimalizačního procesu se však LEM s SVM zvládne skokovitě zlepšit (obr. 5.21 (b) a (c)) a překonává tak kombinovanou variantu.

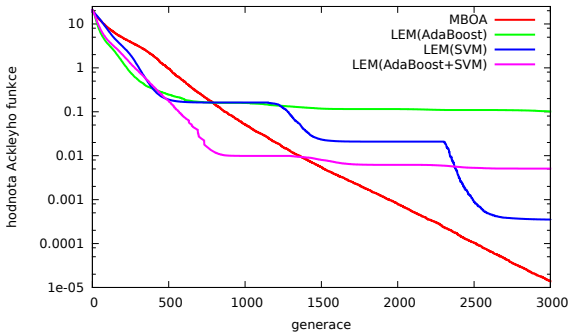
Algoritmus MBOA se k minimu přibližuje stabilně a je celkově nejlepší.



(a) dimenze $N = 10$



(b) dimenze $N = 30$



(c) dimenze $N = 50$

	dimenze		
	10	30	50
MBOA	$6{,}69E{-}6$	**9,84E−7**	**1,39E−5**
LEM(AdaBoost)	$8{,}33E{-}4$	$2{,}5E{-}2$	$1{,}02E{-}1$
LEM(SVM)	**4,96E−6**	$9{,}99E{-}5$	$3{,}51E{-}4$
LEM(AdaBoost+SVM)	$5{,}61E{-}6$	$1{,}87E{-}4$	$5{,}09E{-}3$

(d) dosažené hodnoty

Obrázek 5.21: Vývoj hodnot Ackleyho funkce

5.6.7 Zhodnocení výkonnosti LEM vůči MBOA

Výsledky ukazují, že pomocí metody LEM je možné dosahovat srovnatelných výsledků s EDA algoritmem MBOA ve výrazně kratší době. Avšak velmi záleží na tom, jaký je zvolen klasifikační algoritmus. U sférické a Griewangkovy funkce LEM s SVM výrazně překonává LEM s AdaBoost, u Rastriginovy a Schwefelovy funkce je situace opačná. Tento jev je nejspíše způsoben tím, že LEM(SVM) má problém s překonáním výraznějších hranic lokálního minima a uvázne v něm. LEM(AdaBoost) vytváří více obecné popisy jedinců, tudíž sice konverguje pomaleji, ale lépe se vypořádá s lokálními minimy.

Právě pro řešení tohoto problému byla navržena varianta LEM střídající oba klasifikační algoritmy. A kromě jednoho případu tato kombinovaná varianta vždy překonala LEM(AdaBoost). V případě Rastriginovy a Schwefelovy funkce se vhodně snoubily rozdílné postupy při vytváření jedinců: Přístup dle AdaBoost klasifikátoru přibližně určil oblast globálního minima, pomohl překonat lokální extrémy, následně jedinci vytváření dle SVM pozici minima zpřesnili.

Algoritmus MBOA si vede nejlépe při deseti dimenzích, s jejich počtem bývá překonán některou z variant LEM. Pouze při optimalizaci 30dimenzionální varianty sférické funkce, 10dimenzionální variant Rastriginovy funkce a u třiceti a 50dimenzionální varianty Ackleyho funkce je v dosahované hodnotě lepší. Ale i v těchto případech platí, že do třetiny optimalizační průběhu, konverguje (zejména) kombinovaná varianta LEM rychleji.

Kapitola 6

Experimenty v dynamickém prostředí

Metoda LEM se ukázala být úspěšná při optimalizaci numerických funkcí. Její silnou stránkou je především počáteční rychlá konvergence. Nabízí se tedy ji aplikovat na problémy dynamické, v nichž se extrémy funkce mění v čase. Právě díky rychlé konvergenci by mohla metoda LEM sledovat pohyb optima a být tak úspěšná.

V této kapitole je zkoumáno chování LEM(AdaBoost) a LEM(SVM) na jednom typu dynamického problému – úloze pohybujících se vrcholů. Používaná implementace metod je shodná s implementací pro statické problémy.

6.1 Pohybující se sinusový vrchol

Pro porovnání s EDA algoritmem byla zvolena úloha s pohybujícím se vrcholem v podobě sinusové funkce, na níž byly testovány varianty algoritmu MBOA [18].

Jedná se o funkci s jedním extrémem, který se po určitém počtu generací posune. Její fitness funkce je definována následovně:

$$F(x_1, \ldots, x_N, g) = \frac{1}{N} \sum_{i=1}^{n} f(x_i, g) \tag{6.1}$$

$$f(x, g) = \begin{cases} \sin(x - \lfloor g/T \rfloor \cdot k \ \cdot \pi) & \text{pro } 0 < x - \lfloor g/T \rfloor \cdot k \ \cdot \pi < \pi \\ 0 & \text{jinak} \end{cases} \tag{6.2}$$

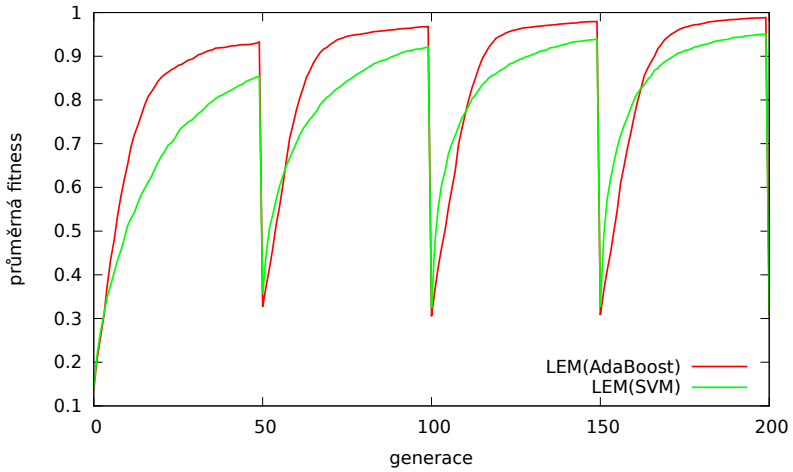
kde g je aktuální generace, k je koeficient posunu vrcholu, T je perioda změny. Maximální hodnota fitness funkce je 1 pro $\forall x_i = \frac{\pi}{2} + \lfloor g/T \rfloor \cdot k \ \cdot \pi$.

Při porovnávání kvality optimalizačních algoritmů na dynamických problémech je vhodné užít jiné metriky. Jednou z nich je kolektivní průměrná hodnota fitness E_c. Je to tzv. offline metrika, která se vypočítá jako průměr nejlepší dosažené hodnoty fitness (F_{BG}) ze všech generací z M opakovaných běhů.

$$E_c = \sum_{g=1}^{G} (\sum_{m=1}^{M} F_{BG}/M)/G \tag{6.3}$$

Pro experimenty je počet opakovaných běhů $M = 100$, počet generací $G = 1000$, koeficient posunu $k \in \{0{,}5, 4\}$. Dimenze problému je $N = 8$ a definiční obor je $< 0, 80\pi >^N$.

Obvyklý průběh optimalizačního procesu u dynamických problémů se vyznačuje prudkými poklesy hodnoty fitness v okamžiku změny prostředí – při posunu vrcholu. Pro LEM(AdaBoost) a LEM(SVM) je zobrazen na obrázku 6.1.



Obrázek 6.1: Fitness nejlepšího jedince v generaci, perioda změny $T = 50$, koeficient změny $k = 0{,}5$, velikost populace 100 jedinců

6.1.1 Vliv velikosti populace na kvalitu řešení

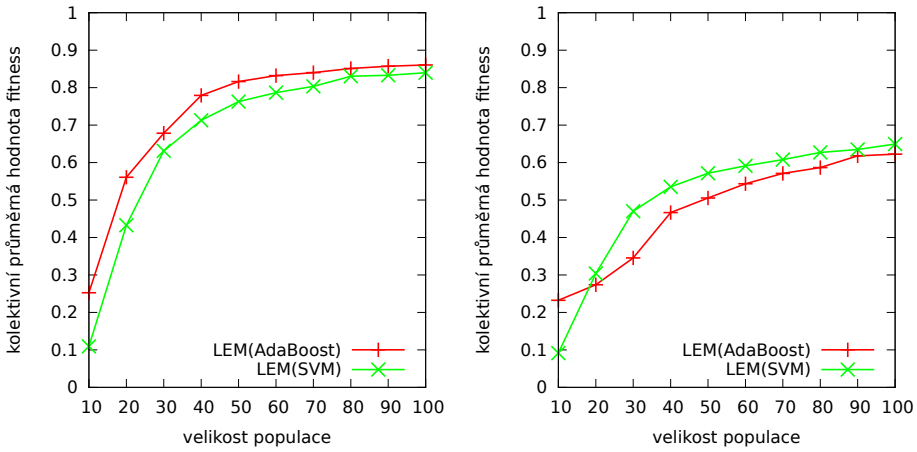
Jelikož zkoumaná funkce má pouze jeden extrém, neměla by metoda mít problém s jeho nalezením. Otázkou tedy je, jak velká populace (tím pádem i počet vyhodnocení fitness funkce) je potřebná k dosažení kvalitního řešení. Samozřejmě se dá předpokládat, že větší populace bude lépe reagovat na změny, na druhou stranu velikost populace přímo souvisí s počtem vyhodnocení fitness funkce a zvláště v dynamických prostředích je důležité, aby tento počet byl co nejnižší.

Pro experiment byla stanovena perioda změny $T = 50$, tedy populace má 50 generací na to, aby nalezla optimum a zvládla se jej držet. Parametry metody LEM byly obdobné jako u předchozích experimentů:

Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	2
Velikost skupiny	10% velikosti populace

Při experimentu byla počáteční velikost populace sto jedinců. Po provedení $M = 100$ běhů je do příslušného grafu v obrázku 6.2 vynesena hodnota kolektivní průměrné fitness. Poté je experiment restartován s populací o deset jedinců větší. Maximální velikost je dvě stě jedinců.

Výsledky experimentu potvrdily, že větší velikost populace přináší lepší výsledky. Nicméně už při populaci o padesáti jedincích nejsou dosahované výsledky výrazně horší než při sto jedincích. Při méně razantním posunu vrcholu (obrázek 6.2(a)) obě varianty LEM reagují na změny v prostředí relativně dobře ($E_c > 0{,}8$), LEM(AdaBoost) překonává LEM(SVM) nezávisle na velikosti populace. Při výraznější změně prostředí (obrázek 6.2(b)) už nejsou dosahované výsledky tak dobré ($E_c > 0{,}5$), tentokrát se lépe posunu přizpůsobuje LEM(SVM).



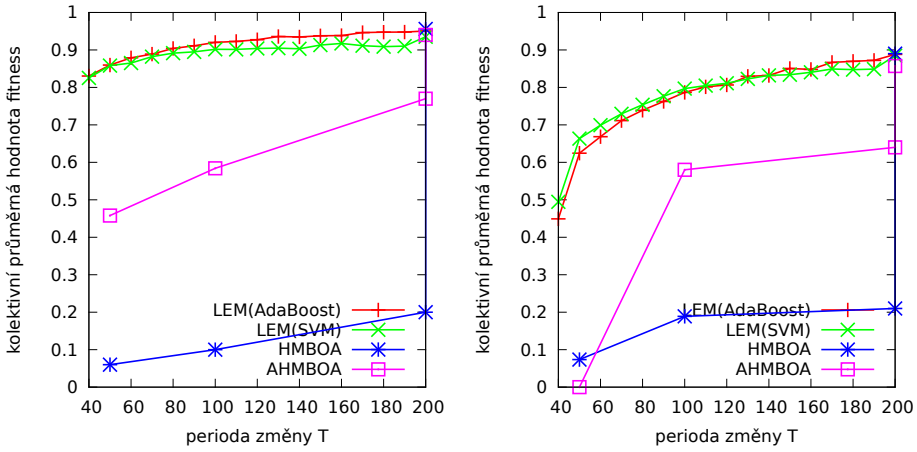
(a) Koeficient posunu $k = 0{,}5$ (b) Koeficient posunu $k = 4{,}0$

Obrázek 6.2: Vzájemné srovnání variant LEM při periodě posunu $T = 50$

6.1.2 Vliv změny periody na kvalitu řešení

Faktory, které ovlivňují dynamiku prostředí, jsou koeficient posunu k a perioda změny T. V tomto experimentu proběhne srovnání výkonnosti při různých periodách změny a zároveň proběhne srovnání s variantami EDA algoritmu MBOA [18].

Parametry metod LEM byly shodné jako v předchozím experimentu, populace čítala sto jedinců. Proběhlo $M = 100$ běhů pro periodu změny $T \in < 40, 200 >$. U MBOA byly publikovány výsledky pro $M = 10$ běhů a pro periodu $T = \{50, 100, 200\}$. Pro porovnání slouží průměrná kolektivní hodnota fitness, výsledky všech variant jsou zaneseny v grafech 6.3.



(a) Koeficient posunu $k = 0{,}5$ (b) Koeficient posunu $k = 4{,}0$

Obrázek 6.3: Srovnání variant LEM s variantami MBOA, pro MBOA při $T = 200$ je dosažena lepší hodnota při použití stráží (Sentinels)

Výsledky ukazují, že malá změna posunu ($k = 0{,}5$, obrázek 6.3(a)) výrazně neovlivňuje hodnotu E_c dosahovanou oběma variantami LEM. LEM(AdaBoost) mírně překonalo LEM(SVM). Co se týče srovnání s MBOA, bez použití stráží obě varianty výrazně ztrácí na LEM, se strážemi dosahují při periodě $T = 200$ takřka stejných výsledků. Pro větší koeficient posunu ($k = 4{,}0$, obrázek 6.3(b)) pro LEM roste hodnota E_c s velikostí periody T od 0,5 k hodnotě 0,9. LEM(SVM) dosahuje mírně lepších výsledků než LEM(AdaBoost) pro $T < 120$, poté je mírně lepší varianta s AdaBoost klasifikátorem. Pro MBOA je situace obdobná jako při $k = 0{,}5$, pouze při použití stráží dosahuje stejné výkonnosti jako LEM.

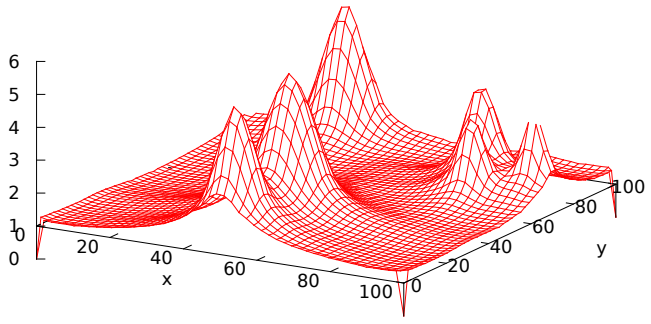
6.2 Úloha s více pohybujícími vrcholy

Jelikož si metoda LEM vedla dobře na předchozím jednoduchém úkolu, rozhodl jsem se ji vyzkoušet v dynamičtějším prostředí s více vrcholy. Pro vytváření takových prostředí byl vytvořen nástroj Moving Peaks Benchmark [19]. Je to často používaný generátor dynamického prostředí, na němž se testují kvality optimalizačních algoritmů.

Nástroj má širokou škálu parametrů, jež ovlivňují podobu prostředí – zejména počet a tvar vrcholů, velikosti změn a rychlost změn prostředí.

Jak je vidět na obrázku 6.4, mimo vrcholy je hodnota fitness funkce nulová. Optimalizační průběh se dá tedy rozložit do dvou fází. Nejdříve se populace musí dostat z ploché části, následně musí sledovat pohyb vrcholu. Jiným problémem je, když se populace upne na nesprávný vrchol. V takovém případě je obtížné přejít na jiný vrchol právě kvůli ploché části, nejsnadnějším řešením je v tom případě časté náhodné generování jedinců.

Nástroj poskytuje metriku hodnotící kvalitu optimalizace – je to tzv. offline chyba, která udává průměr rozdílů nejlepší dosažené hodnoty od nejlepší hodnoty v prostředí ze všech vyhodnocení fitness funkce.



Obrázek 6.4: Prostředí s několika vrcholy

6.2.1 Vliv počtu skupin a jejich velikosti na kvalitu řešení

Cílem experimentu bylo zjistit, jestli vytváření jedinců z více skupin nepřinese lepší výsledky ve smyslu menší offline chyby. Důvodem, proč by tomu tak mohlo být, je fakt, že při změně prostředí se může vrchol posunout na pozici, která je pokryta jedincem, který před změnou nebyl ve skupině nejlepších.

Pro Moving Peak Benchmark jsou uváděny doporučené konfigurace, tzv. scénáře. V experimentu byl použit scénář 2 s parametry:

Počet vrcholů	10
Dimenze problému	5
Definiční obor	$< 0, 100 >$
Výška vrcholu	$< 30, 70 >$
Šířka vrcholu	$< 1, 12 >$
Změna pozice o	1.0
Změna výšky o	1.0
Změna šířky o	0.01
Změna prostředí po	5000 vyhodnoceních
Počet vyhodnocení	500000

Parametry metod LEM:

Velikost populace	100 jedinců
Počet jedinců generovaných z klasifikátoru	70% velikosti populace
Počet jedinců generovaných evolučně	20% velikosti populace
Počet jedinců generovaných náhodně	10% velikosti populace
Počet skupin	$2, 3, 4, 5$
Velikost skupin	$5, 10, 15, 20$ jedinců

Pro každou z možností počtu a velikosti skupin se experiment opakoval stokrát. Průměr z dosažené offline chyby je uveden v tabulce 6.1.

velikost skupiny 5 jedinců	Počet skupin			
	2	3	4	5
LEM(AdaBoost)	8,53	8,01	6,55	6,05
LEM(SVM)	5,08	6,8	5,98	4,6
LEM(AdaBoost+SVM)	5,13	5,11	5,0	5,6

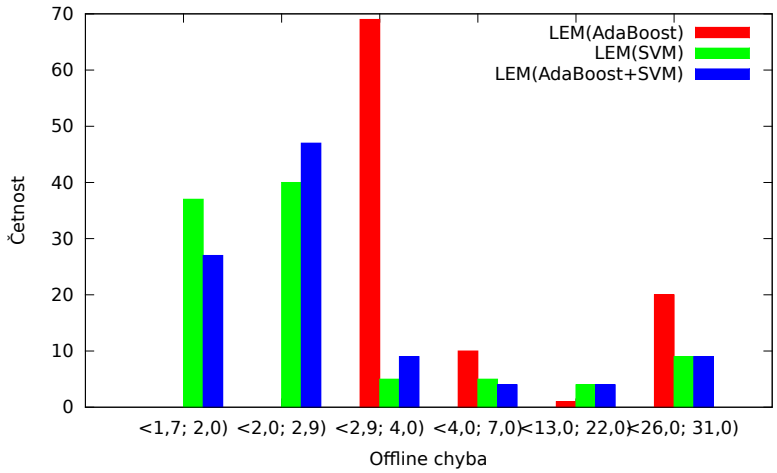
velikost skupiny 10 jedinců	Počet skupin			
	2	3	4	5
LEM(AdaBoost)	8,08	7,27	6,62	7,09
LEM(SVM)	6,37	5,6	4,81	6,39
LEM(AdaBoost+SVM)	7,46	5,97	4,79	6,02

velikost skupiny 15 jedinců	Počet skupin			
	2	3	4	5
LEM(AdaBoost)	5,81	9,57	7,67	7,62
LEM(SVM)	5,88	4,78	6,4	6,2
LEM(AdaBoost+SVM)	5,74	5,2	5,14	5,75

velikost skupiny 20 jedinců	Počet skupin			
	2	3	4	5
LEM(AdaBoost)	6,87	7,99	8,24	9,02
LEM(SVM)	6,39	5,43	6,54	6,59
LEM(AdaBoost+SVM)	4,93	4,61	6,15	4,94

Tabulka 6.1: Offline chyba při různém počtu skupin

Největším problémem při vyhodnocení výsledků byl fakt, že metoda ne vždy nalezla vrchol odpovídající globálnímu maximu. Průměrné hodnoty jsou tedy silně zatíženy případy, kdy populace sledovala nesprávný vrchol. Histogram dosahovaných hodnot pro dvě skupiny o pěti jedincích (obrázek 6.5) ukazuje, že ve třetině optimalizačních běhů LEM(SVM) byla hodnota offline chyby menší než 2, osmdesát běhů skončilo s offline chybou menší než 4, nicméně několik dosáhlo hodnoty chyby větší než 26.



Obrázek 6.5: Četnost dosahovaných hodnot ze sta běhů, při použití dvou skupin o pěti jedincích

Výsledky nedávají jednoznačnou odpověď, zda je výhodné používat více skupin. Nejmenší hodnoty offline chyby bylo sice dosaženo pro pět skupin o pěti jedincích u LEM s SVM klasifikátorem, ale například pro čtyři skupiny byla výsledná hodnota horší než pro skupiny dvě. Jediným případem, kdy s počtem skupin klesala offline chyba, byla optimalizace metodou LEM(AdaBoost) s pěti jedinci ve skupině.

Kapitola 7

Závěr

Záměrem mojí diplomové práce bylo navrhnout a implementovat novou variantu evolučního modelu s učením, která by dosahovala kvalitních výsledků při nízkém počtu vyhodnocení fitness funkce. Za tímto účelem jsem navrhl dva přístupy tvorby nových jedinců založené na úspěšných klasifikačních algoritmech AdaBoost a SVM. Další modifikací oproti dřívějším verzím LEM je využívání více výkonnostních skupin pro tvorbu nových generací. Během provádění experimentů byly zjištěny výrazné rozdíly ve výkonnosti metody v závislosti na použitém klasifikačním algoritmu. Z toho důvodu jsem vytvořil třetí variantu LEM střídající obě klasifikační metody.

Výsledky experimentů ve statickém prostředí poukázaly na řadu vlastností nových verzí LEM. Více než vhodné je kombinovat běh metody s evolučními operátory křížení a mutace, bez nich jsou dosahované výsledky znatelně horší. Dalším důležitým zjištěním bylo, že metody dosahují lepších výsledků při relativně malých velikostech skupin. Zatímco v ostatních pracích bylo doporučováno využívat 30% populace na jednu skupinu, pro LEM s klasifikátorem AdaBoost i LEM s SVM bylo lepších výsledků dosahováno s pouhými 5% populace pro každou ze skupin. Využívání více skupin pro tvorbu nových jedinců nepřineslo očekávané zlepšení. Rozdíly v dosahovaných hodnotách byly znatelné především v několika prvních generacích, v pozdějších fázích evolučního procesu byly odchylky minimální.

Srovnání s původní metodou LEM3 ukázalo na nevýhody použití diskretizace. Pokud byly testovací funkce náhodně posunuty, pak metoda LEM3 v konvergenci velmi rychle ustrnula a celkově dosahovala horších hodnot než LEM s klasifikátorem AdaBoost. Avšak při ponechání funkcí na původních pozicích metoda LEM3 našla globální optimum během několika generací.

Porovnání s EDA algoritmem MBOA probíhalo v 10, 30 a 50dimenzionálních variantách testovacích funkcí. LEM kombinující AdaBoost s SVM dosahovala srovnatelných či lepších hodnot než MBOA rychleji. V několika případech však byla ke konci optimalizačního procesu překonána algoritmem MBOA. Ten obecně konvergoval sice pomaleji, ale stabilně ke globálnímu optimu. Nejtěžším úkolem bylo hledání minima Rosenbrockovy funkce, při kterém všechny porovnávané algoritmy selhaly.

V dynamickém prostředí si metoda LEM nevedla tak dobře jako v neměnném prostředí. Srovnání s variantami EDA algoritmu MBOA pro dynamické úlohy proběhlo na problému sledování pohybu jednoho vrcholu. Metody LEM dosahovaly lepších hodnot než varianty MBOA, srovnatelných hodnot dosáhly při nejmenší periodě pohybu vrcholu.

Vliv velikosti a počtu skupin byl zkoumán na úloze s více pohyblivými vrcholy. Stejně jako ve statickém prostředí dosahovaly lepších výsledků menší velikosti skupin, počet skupin spíše neměl vliv na kvalitu řešení. Největším problémem v dynamickém prostředí bylo

uváznutí v lokálním maximu. Z celkového pohledu si v méně dynamickém prostředí vedla lépe metoda LEM s klasifikátorem AdaBoost. Jakmile však byly změny výraznější, tak se LEM s klasifikátorem SVM zvládla přizpůsobovat rychleji.

Jelikož metoda LEM byla zkoušena pouze na testovacích funkcích, tak další možný vývoj práce vidím zejména ve smyslu aplikace metody LEM na řešení reálných problémů. Především v optimalizačních problémech, pro které je vyhodnocení jejich fitness funkce časově náročné. Metoda LEM by tu mohla uspět díky velmi rychlé konvergenci. Vhodným rozšířením by také byla úprava metody pro řešení jiných typů optimalizačních problémů, například z oblasti multikriteriální optimalizace.

Literatura

[1] Michalski, R. S. Learnable Evolution: Combining Symbolic and Evolutionary Learning. In *Proceedings of the Fourth International Workshop on Multistrategy Learning (MSL'98)* [online]. 1998 [cit. 27. září 2013]. S. 14–20. Dostupné na: http://www.mli.gmu.edu/papers/96-2000/98-09.pdf.

[2] Cervone, G. *LEM2: Theory and Implementation of the Learnable Evolution Model* [online]. 1999 [cit. 5. listopadu 2013]. Dostupné na: http://www.mli.gmu.edu/papers/96-2000/99-6.pdf.

[3] Michalski, R. S., Esposito, F. a Saitta, L. Learnable evolution model: Evolutionary processes guided by machine learning. In *Machine Learning* [online]. 2000 [cit. 27. září 2013]. S. 9–40. Dostupné na: http://www.mli.gmu.edu/papers/96-2000/00-2.pdf.

[4] Wojtusiak, J. a Michalski, R. S. The LEM3 System for Non-Darwinian Evolutionary Computation and Its Application to Complex Function Optimization. In *George Mason University* [online]. 2005 [cit. 5. listopadu 2013]. Dostupné na: http://www.mli.gmu.edu/papers/2005/05-5.pdf.

[5] Cervone, G. a Michalski, R. S. *Adaptive Anchoring Discretization for Learnable Evolution Model: The ANCHOR Method* [online]. 2001 [cit. 27. prosince 2013]. Dostupné na: http://digilib.gmu.edu/dspace/handle/1920/1473.

[6] Coletti, M., Lash, T., Mandsager, C. et al. *An Experimental Application of Learnable Evolution Model and Genetic Algorithms to Parameter Estimation in Digital Signal Filters Design* [online]. 1999 [cit. 4. října 2013]. Dostupné na: http://www.mli.gmu.edu/papers/96-2000/99-5.pdf.

[7] Cervone, G., Kaufman, K. A. a Michalski, R. S. Validating Learnable Evolution Model on Selected Optimization and Design Problems. In *George Mason University* [online]. 2003 [cit. 11. listopadu 2013]. Dostupné na: http://www.mli.gmu.edu/papers/2003-2004/mli03-1.pdf.

[8] Michalski, R. S. a Zhang, Q. Initial Experiments with the LEM1 Learnable Evolution Model: An Application to Function Optimization and Evolvable Hardware. In *George Mason University* [online]. 1999 [cit. 4. října 2013]. Dostupné na: http://digilib.gmu.edu/dspace/bitstream/1920/1858/1/99-04.pdf.

[9] Sheri, G. a Corne, D. The simplest evolution/learning hybrid: LEM with KNN. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on* [online]. 2008 [cit. 14. října 2013]. S. 3244–3251. Dostupné na: http://www.macs.hw.ac.uk/~dwcorne/LEMfinal.pdf.

[10] Burgetová, I. *Klasifikace a predikce.* Brno: FIT VUT v Brně, 2013. Přednáška z předmětu ZZN.

[11] Sheri, G. a Corne, D. *Evolutionary Optimization Guided by Entropy-Based Discretization* [online]. 2009 [cit. 21. prosince 2013]. 695-704 s. Dostupné na: http://dx.doi.org/10.1007/978-3-642-01129-0_79.

[12] Shehab, M. E., Badran, K. a Salama, G. I. A Generic Feature Extraction Model using Learnable Evolution Models (LEM+ID3). *International Journal of Computer Applications* [online]. February 2013, roč. 64, č. 11 [cit. 14. října 2013]. S. 27–32. Published by Foundation of Computer Science, New York, USA.

[13] Sheri, G. a Corne, D. Learning-assisted evolutionary search for scalable function optimization: LEM(ID3). In *Evolutionary Computation (CEC), 2010 IEEE Congress on* [online]. 2010 [cit. 14. října 2013]. S. 1–8.

[14] Jourdan, L., Corne, D., Savic, D. et al. *Preliminary Investigation of the Learnable Evolution Model for Faster/Better Multiobjective Water Systems Design* [online]. 2005 [cit. 14. října 2013]. Dostupné na: http://www.lania.mx/~ccoello/EMOO/jourdan05.pdf.gz.

[15] Weiss, M. *Evoluční model s učením (LEM) pro optimalizační úlohy.* Brno: Vysoké učení technické, Fakulta informačních technologií, 2011. Diplomová práce. Dostupné na: http://www.fit.vutbr.cz/study/DP/DP.php?id=10396.

[16] Platt, J. C. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines* [online]. 1998 [cit. 5. dubna 2014]. Dostupné na: http://research.microsoft.com/en-us/um/people/jplatt/smoTR.pdf.

[17] Očenášek, J. *Paralelní evoluční algoritmy využívající pravděpodobnostní modely.* Brno: Vysoké učení technické, Fakulta informačních technologií, 2002. Disertační práce. Dostupné na: http://jiri.ocenasek.com/papers/ocenasek_phd.pdf.

[18] Kobliha, M., Schwarz, J. a Očenášek, J. Bayesian Optimization Algorithms for Dynamic Problems. In Rothlauf, F., Branke, J., Cagnoni, S. et al. (ed.). *Applications of Evolutionary Computing* [online]. [b.m.]: Springer Berlin Heidelberg, 2006 [cit. 8. dubna 2014]. S. 800–804. Lecture Notes in Computer Science, sv. 3907. Dostupné na: http://dx.doi.org/10.1007/11732242_77. ISBN 978-3-540-33237-4.

[19] Branke, J. *Moving Peaks Benchmark* [online]. 1999 [cit. 7. února 2013]. Dostupné na: http://people.aifb.kit.edu/jbr/MovPeaks/.

Příloha A

Obsah DVD

Přiložený disk obsahuje:

- Zdrojové kódy implementací metod LEM
- Zdrojové kódy testovacích programů pro statické a dynamické prostředí
- Naměřená data z experimentů použitá pro generování grafů
- Text diplomové práce ve formátu pdf a LaTeX

Příloha B

Manuál

V adresáři `/implementace` na přiloženém DVD se nacházejí zdrojové kódy LEM a testovacích programů. Po spuštění nástroje `make` se dle souboru `Makefile` vytvoří následující spustitelné programy (pro přeložení se vyžaduje překladač C++ normy C++11 – testováno s `g++` verze 4.7):

- `lem{AdaBoost, SVM, Combined}` – Program pro testování ve statickém prostředí. Výstupem programu jsou dva sloupce, první představuje nejlepší dosaženou hodnotu, druhý průměrnou dosaženou hodnotu v populaci. Řádky odpovídají generacím. Volitelné parametry jsou:

Parametr	možné hodnoty	výchozí hodnota	popis
`-f`	sphere, griewangk, rosenbrock, rastrigin, schwefel, ackley	sphere	funkce k optimalizaci
`-d`	kladná celá čísla	10	počet dimenzí
`-g`	kladná celá čísla	100	počet generací
`-p`	kladná celá čísla	100	velikost populace
`-l`	(0,0, 1,0)	0,7	jaká část populace se vygeneruje dle klasifikátoru
`-e`	(0,0, 1,0)	0,2	jaká část populace se vygeneruje evolučně
`-lb/-ub`	reálná čísla	$-600/600$	spodní/horní hranice definičního oboru
`-grs`	kladná celá čísla	2	počet skupin
`-mg`	(0,0, 1,0)	0.05	velikost skupiny v závislosti na velikosti populace

- `movingSin{AdaBoost, SVM, Combined}` – Program pro úlohu pohybujícího se sinusového vrcholu dle [18]. Výstupem je kolektivní průměrná hodnota fitness. Parametry shodné s `lem` bez `-f`, navíc přibyly (nebo mají jiné výchozí hodnoty):

`-d`	kladná celá čísla	8	počet dimenzí
`-g`	kladná celá čísla	1000	počet generací
`-lb/-ub`	reálná čísla	$0/80\pi$	spodní/horní hranice definičního oboru
`-T`	kladná celá čísla	50	Perioda změny
`-M`	kladná celá čísla	10	Počet opakovaných běhů
`-k`	reálná čísla	0.5	koeficient posunu

- `movingPeaks{AdaBoost, SVM, Combined}` – Program pro Moving Peak Benchmark. Výstupem je offline chyba. Volitelné parametry jsou shodné s `lem` bez `-f`, navíc přibyly nebo jsou upraveny:

`-d`	kladná celá čísla	5	počet dimenzí
`-g`	kladná celá čísla	5000	počet generací
`-lb/-ub`	reálná čísla	0/100	spodní/horní hranice definičního oboru
`-freq`	kladná celá čísla	5000	Počet vyhodnocení fitness pro změnu vrcholů
`-peaks`	kladná celá čísla	10	Počet vrcholů

Metoda LEM byla implementována tak, aby byla snadno použitelná v rámci jakéhokoli programu. Rozhraní třídy metody je v hlavičkovém souboru `Lem.h`. Konstruktoru třídy je možné/nutné zadat:

Parametr	možné hodnoty	výchozí hodnota
`Fitness funkce`	pracuje nad vektorem reálných čísel	žádná
`Vzor jedince`	Vektor dvojic spodní a horní hranice definičního oboru pro každou dimenzi	žádná
`Velikost populace`	kladné celé číslo	100
`Poměr jedinců generovaných učením`	(0,0, 1,0)	0,7
`Poměr jedinců generovaných evolučně`	(0,0, 1,0)	0,2
`Počet skupin`	kladné celé číslo	2
`Poměrná velikost skupiny`	(0,0, 1,0)	0,05
`Hledá se`	maximum/minimum fitness funkce	maximum
`Typ klasifikátoru`	AdaBoost, SVM, Kombinace	AdaBoost

Metody třídy jsou zejména:

- `run(počet generací)` – Provede zadaný počet generací.
- `getStatistic()` – Vrací strukturu obsahující vektor nejlepších hodnot v populaci, vektor průměrných hodnot populace a vektor celkově nejlepších řešení.