

# Identifikace proměnných hvězd pomocí evolučních technik

Identification of Variable Stars using Evolutionary Techniques

Bc. Vít Hrbáček

---

Diplomová práce  
2012



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2011/2012

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Vít Hrbáček**  
Osobní číslo: **A10467**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Počítačové a komunikační systémy**  
Forma studia: **prezenční**

Téma práce: **Identifikace proměnných hvězd pomocí evolučních technik**

Zásady pro vypracování:

1. Vypracujte rešerši zabývající se problematikou identifikace proměnných hvězd, včetně používaných softwarových produktů.
2. Navrhněte vlastní softwarové prostředí.
3. Navrhněte a implementujte evoluční techniky pro vyhledávání ideálních kombinací referenčních hvězd.
4. Aplikujte na dodané datové soubory.
5. Získané výsledky zpracujte formou grafické a tabelární formy, které budou součástí závěru.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. ZELINKA, Ivan, OPLATKOVÁ Zuzana, ŠEDA Miloš, OŠMERA Pavel a VČELAŘ František. Evoluční výpočetní techniky: principy a aplikace. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.
2. OPLATKOVÁ, Zuzana. Analytic programming. Zlín, 2003. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Vedoucí práce prof. Ing. Ivan Zelinka, Ph.D.
3. HYNEK, Josef. Genetické algoritmy a genetické programování. 1. vyd. Praha: Grada, 2008. ISBN 978-80-247-2695-3 (brož.).
4. ZELINKA, Ivan. Evolutionary algorithms and chaotic systems. Berlin : Springer, 2010. 521 s. ISBN 978-3-642-10707-8.
5. Kvasnička V., Pospíchal J., Tiňo P., Evolučné algoritmy, STU Bratislava, 2000, 215 s. ISBN 80-227-1377-5
6. TVRDÍK, Josef. Evoluční algoritmy. Ostrava, 2004. Dostupné z: [http://prf.osu.cz/doktorske\\_studium/dokumenty/Evolutionary\\_Algorithms.pdf](http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf). Učební text. Ostravská univerzita.

Vedoucí diplomové práce:

**prof. Ing. Ivan Zelinka, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

**28. července 2012**

Termín odevzdání diplomové práce:

**4. září 2012**

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.

*děkan*



prof. Ing. Karel Vlček, CSc.

*ředitel ústavu*

## **ABSTRAKT**

Tato diplomová práce se zabývá využitím evolučně výpočetních technik (EVT) při identifikaci proměnných hvězd. V teoretické části je popsáno základní dogma EVT, rozdělení evolučních výpočetních technik a jejich stručný popis funkcionality základních z nich. Poté se práce zabývá rozdělením a popisem proměnných hvězd a soustav hvězd, jejich pozorováním a měření. Ve druhé, praktické části je popsán postup při návrhu programového řešení pro nahrazení manuálního výpočtu.

Klíčová slova: Evoluční výpočetní techniky, EVT, Evoluční algoritmy, jedinec, populace, generace, vhodnost, proměnné hvězdy, CCD, Mathematica.

## **ABSTRACT**

This diploma thesis deals the use of evolutionary computing (EVT) to identification of variable stars. First is described the basic dogma of EVT, a division of evolutionary computing and a brief description of the basic functionality of them in the theoretical part of the thesis. Then the thesis deals with the distribution and description of variable stars and star systems, their observation and measurement. The design of software solutions to replace manual calculation is described in the second part of the thesis.

Keywords: Evolutionary Computing, EVT, Evolutionary Algorithms, Individual, Population, Generation, Fitness, Variable Stars, CCD, Mathematica.

Děkuji panu profesorovi Ing. Ivanu Zelinkovi, Ph.D., vedoucímu diplomové práce, za odbornou pomoc, cenné rady, připomínky a veškerý strávený čas při tvorbě této práce. Rodině za poskytnuté zázemí a podporu při studiu.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji, že**

- jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>11</b>
<b>1 ÚVOD DO EVOLUČNÍCH VÝPOČETNÍCH TECHNIK</b> .....	<b>12</b>
1.1 ZÁKLADNÍ DOGMA EVT.....	12
1.2 ROZDĚLENÍ EVT.....	14
1.3 ZÁKLADNÍ POJMY.....	16
<b>2 VYBRANÉ OPTIMALIZAČNÍ A EVOLUČNÍ TECHNIKY</b> .....	<b>18</b>
2.1 SLEPÝ ALGORITMUS.....	18
2.2 HOROLEZECKÝ ALGORITMUS.....	19
2.3 SIMULOVANÉ ŽIHÁNÍ.....	21
2.4 GENETICKÉ ALGORITMY.....	23
2.4.1 Výběr rodičů.....	24
2.4.2 Křížení a mutace.....	24
2.5 EVOLUČNÍ STRATEGIE.....	26
2.5.1 Dvoučlenné evoluční strategie.....	27
2.5.2 Vícečlenné evoluční strategie.....	28
2.5.3 Rekombinační evoluční strategie.....	28
2.6 ACO.....	29
2.7 SOMA.....	30
2.7.1 Křížení a mutace.....	31
2.8 DIFERENCIÁLNÍ EVOLUCE.....	32
2.8.1 Mutace a křížení.....	33
2.9 NO FREE LUNCH TEORÉM.....	35
<b>3 IDENTIFIKACE PROMĚNNÝCH HVĚZD</b> .....	<b>36</b>
3.1 PROMĚNNÉ HVĚZDY.....	36
3.1.1 Geometrické proměnné hvězdy.....	36
3.1.2 Fyzické proměnné hvězdy.....	37
3.2 POZOROVÁNÍ A MĚŘENÍ HVĚZD.....	42
<b>II PRAKTICKÁ ČÁST</b> .....	<b>46</b>
<b>4 NÁVRH PROGRAMOVÉHO ŘEŠENÍ</b> .....	<b>47</b>
4.1 PŘEVEDENÍ *.MAT NA *.TXT.....	48
4.2 ZPRŮMĚROVÁNÍ JASŮ.....	50
4.3 SEŘAZENÍ PRŮMĚRŮ DLE $\Delta$ .....	53
4.4 NOMINÁLNÍ HVĚZDA.....	55
4.5 APLIKACE EVOLUCE.....	57

---

4.5.1	SOMA .....	58
4.5.2	Diferenciální evoluce .....	61
<b>ZÁVĚR.....</b>		<b>67</b>
<b>CONCLUSION.....</b>		<b>68</b>
<b>SEZNAM POUŽITÉ LITERATURY .....</b>		<b>69</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>		<b>70</b>
<b>SEZNAM OBRÁZKŮ .....</b>		<b>74</b>
<b>SEZNAM TABULEK .....</b>		<b>76</b>

## ÚVOD

Byla tu již od vzniku planety Země, a ačkoli je tak „stará“, stále je aktuální a velice zajímavá. Nejedná se o nic jiného, než o samotnou *evoluci*. Evoluce lidstva, ale i zvířat, rostlin a planety vůbec zajišťuje vývin, rozvoj nových druhů a to jak křížením, tak mutací, degenerací či zánikem. Základním stavebním prvkem evoluce je chromozóm, který se skládá z genů. Křížením rodičů se předávají určité geny potomkům, vzniká nová populace, nový rod, křížením rodičů odlišných druhů vzniká nový druh. Obecné předpoklady říkají, že pro vznik nové, lepší populace je zapotřebí, aby vybraní rodiče pro křížení byli nejlepší z populace, ovšem občas je zapotřebí, byl přijat i horší jedinec a tím nedošlo k přešlechtění „degeneraci“ populace. Geny mohou podléhat mutacím, ať už k lepšímu, či k horšímu a opět vzniká a vyvíjí se „nový“ druh. Zánikem druhu vzniká prostor pro vývin, rozvoj druhu jiného. Evoluce je všemocná a nevyzpytatelná.

Jako první si určitých zákonů evoluce všiml Jean Baptiste Lamarck, který vypracoval *první evoluční teorii*. Ovšem dnešní zákony evoluce vychází z později předložených teorií známějších a uznávanějších pánů Charlese Roberta Darwina a Gregora Johanna Mendela. Tyto teorie se také později staly řídicími pro evoluční výpočetní techniky a zvláště genetické algoritmy. Ačkoli se o evoluční teorii zasloužili právě Darwin a Mendele, dle (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) sahají kořeny evoluční teorie až k mysliteli Anaximandru z Milétu (přelom 7. – 6. stol. př. n. l.), který své myšlenky zaznamenal ve svém spise *O přírodě*, z něhož se ale zachovaly pouze zlomky. Mezi prvními, kdo využili evoluční teorie v programování, přesněji v optimalizaci byl Rechenberg (Rechenberg, 1973) při návrhu evoluční strategie.

Evoluci si „vymyslela“ příroda při vzniku planety Země k vývoji a rozvoji živočišných druhů. Lidé princip evoluce prozkoumali, popsali teorii evoluce a začali využívat principů evoluce v nejrůznějších oborech, ať už pro maximalizaci zisku, či minimalizaci nákladů, spotřebovaného materiálu, času, pro získání nejlepších konstrukčních rozměrů, nejkvalitnějších a nejlepších kombinací a mnoho dalších. O nejrůznějších užitích evolučních výpočetních technik se lze více dočíst v (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009), (Kvasnička, Pospíchal, Tiňo, 2000) nebo (Hynek, 2008).

Nyní se dostáváme k myšlence, která vedla ke vzniku této práce. Proč nevyužít evolučních algoritmů také v oblasti astronomie? Konkrétně pro identifikaci proměnných hvězd za pomoci evolučních výpočetních technik. Jistě je řada softwarových produktů, které dokážou za pomoci nejrůznějších algoritmů z naměřených dat identifikovat proměnnou hvězdu, nýbrž cílem této práce je využít algoritmů evolučních pro vyhledání nejkvalitnější kombinace hvězd, tzv. *referenčních*, které by sloužily pro porovnání proměnlivosti hvězdy. K tomuto vznikla tato práce, kde v první části jsou popsány základní optimalizační algoritmy a jejich rozdělení, čtenář je také seznámen se základními pojmy optimalizace, jako je např., co to je účelová funkce, ohodnocení účelové funkce, resp. vhodnost. Co to je populace, že se skládá z jedinců a další. Protože smyslem práce bylo využití evolučních technik v astronomii, je čtenář také obeznámen s problematikou proměnných hvězd, jejich rozdělením a jejich pozorováním, resp. s přístroji použitelnými k pozorování a měření hvězd, obecně objektů oblohy. Ve druhé části této práce je pak popsán postup při návrh programového řešení, použitým na dodaných datových souborech.

## **I. TEORETICKÁ ČÁST**

# 1 ÚVOD DO EVOLUČNÍCH VÝPOČETNÍCH TECHNIK

## 1.1 Základní dogma EVT

Tak jak příroda se řídí určitými pravidly, které byly popsány pány Charlesem Robertem Darwinem, Gregorem Johannem Mendelem a jejich teorií evoluce, tak i *Evoluční výpočetní techniky (EVT)* využívají k dosažení velmi dobrých výsledků zákony evoluce, jako jsou dědičnost, křížení, mutace atd.

Základní cyklus každého evolučního algoritmu je možné popsat několika základními kroky:

### 1. Definice parametrů

Před samotným během algoritmu musí být nadefinovány parametry jak řídicí, jako je například *hodnota účelové funkce (cost function)*, případně upravená hodnota účelové funkce, zpravidla do intervalu  $(0, 1)$ , nazývána *vhodnost (fitness)*, tak ukončovací, které regulérně ukončí běh algoritmu. Mezi ukončovací parametry patří např. počet *cyklů (generací, migrací)*, u algoritmu SOMA pak parametr *MinDiv (Minimal Diversity)*, dle starší terminologie jako *AcceptedError*, který udává maximální povolený rozdíl mezi nejlepším a nejleším jedincem v aktuální populaci.

### 2. Prvotní populace (rodiče)

Běh algoritmu začíná vygenerováním prvotní populace jedinců, tzv. rodičů. Jedincem se rozumí vektor, který obsahuje tolik čísel, kolik argumentů je optimalizováno v účelové funkci, obecně označováno písmenem  $N$  (někdy též  $D$ ). Počet jedinců v populaci je označován písmenem  $M$ . Populace je tedy množina jedinců, matice  $N \times M$ . Každý jedinec je jeden z možných řešení daného problému.

### 3. Ohodnocení jedinců (rodičů)

Každý jedinec, rodič, se ohodnotí přes účelovou funkci daného problému a vrátí hodnotu účelové funkce, případně upravenou hodnotu účelové funkce, zpravidla do intervalu  $(0, 1)$ , nazývanou vhodnost.

#### 4. Výběr rodičů

Následně se dle ohodnocení vyberou ti nejlepší potomci, rodičové, ze kterých se dále bude vytvářet nová populace potomků.

#### 5. Nová populace (potomci)

Tato nová populace potomků je tvořena z vybraných rodičů v kroku 4 a to *křížením* daných rodičů. Proces křížení probíhá u každého algoritmu odlišně.

#### 6. Mutace

Tak jak v přírodě nedochází jen ke křížení genů, ale i k jejich mutaci, tak i jedinci u evolučních algoritmů podléhají jistým *mutacím*. Tato mutace se projevuje, jako náhodná změna některých parametrů jedince pomocí náhodného procesu.

Pozn.: je-li proces mutace 100%, mění se i vhodně zvolený algoritmus na pouhé náhodné prohledávání.

#### 7. Ohodnocení jedinců (potomků)

V tomto kroku se postupuje stejně, jako v kroku 3, pouze s tím rozdílem, že tentokrát jsou ohodnocováni potomci.

#### 8. Výběr jedinců (z potomků + rodičů) do nové populace

Z potomků, někdy i ze stávajících nejlepších rodičů, se vyberou nejlepší jedinci a ti naplní novou populaci pro další evoluční cyklus.

#### 9. Nová populace za starou

Tato nová populace nahradí starou populaci a pokračuje se opět krokem 4.

Kroky 4 – 9 se opakují do té chvíle, dokud není splněna některá s ukončovacích podmínek, jako např. vyčerpání zadaných evolučních cyklů (migrací, generací), nebo splnění potřebné kvality výsledku řešení (parametr MinDiv u algoritmu SOMA). Jak již bylo zmíněno, tímto cyklem (kroky 1 – 9) se řídí evoluční algoritmy, algoritmy odlišující se od tohoto předpisu se neřadí algoritmů evolučních, ale k algoritmům spadajících do

EVT. Příkladem může být právě optimalizace mravenčí kolonií (*ACO – Ant Colony Optimization*)

## 1.2 Rozdělení EVT

Evoluční výpočetní techniky, neboli, tzv. *Evoluční algoritmy (Evolutionary algorithms)*, jsou optimalizační algoritmy spadající do skupiny heuristických algoritmů. Ty můžeme dále rozdělit na *deterministické* a *stochastické*.

Evoluční algoritmy lze dělit podle hodně kritérií, např. dle strategie řešení problémů můžeme dělit do dvou skupin, a to:

a) Metody založené na **bodové strategii**.

Princip této metody je ve vyhledávání nového lepšího řešení v sousedství aktuálního nejlepšího řešení. K těmto algoritmům můžeme řadit např. *horolezecký algoritmus*, *simulované žihání* nebo *zakázané prohledávání*.

b) Metody založené na **strategii populace**.

Principem této metody není práce pouze s jedním bodem v sousedství aktuálního řešení, ale s množinou bodů, tzv. *populací*. K těmto algoritmům patří např. *genetické algoritmy* nebo *SOMA*. U algoritmu *ACO* se populace nazývá *kolonie*.

Dále můžeme algoritmy dělit do čtyř skupin dle vlastností řešeného problému, schematické rozdělení lze vidět na obrázku (Obr. 1-1):

a) **Enumerativní**

Tyto algoritmy provedou výpočet všech možných řešení problému a následně je vybrán nejlepší z nich. Problém těchto algoritmů, jak již z popisu vyplívá, je jejich časová náročnost. Jsou vhodné pouze pro řešení problémů s diskretními argumenty účelové funkce, jež nabývají jen velmi malého množství hodnot, tzn. je prohledáván velmi malý prostor možných řešení.

**b) Deterministické**

Algoritmy spadající do této kategorie jsou vhodné při splnění několika omezujících podmínek na daný problém. Jsou-li splněny tyto podmínky, pak jsou tyto algoritmy rychlé a efektivní. Tyto podmínky obvykle jsou:

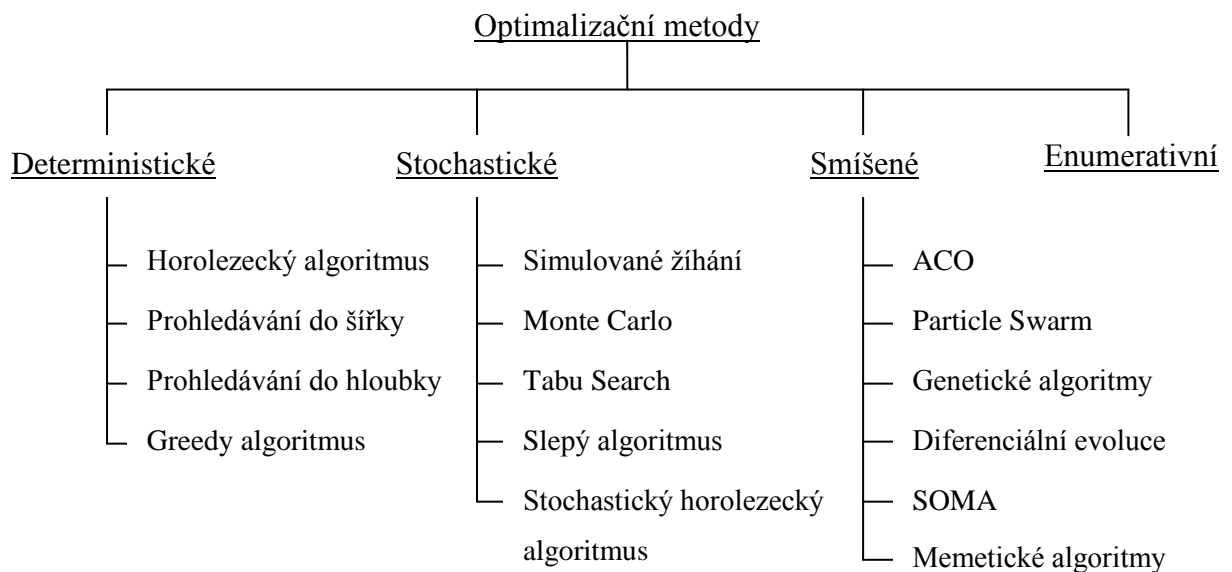
- Řešený problém je lineární a konvexní.
- Prostor možných řešení problému je malý a souvislý.
- Účelová funkce je nejlépe unimodální, tzn., má pouze jeden extrém.
- Řešený problém je definován v analytickém tvaru.
- Mezi parametry účelové funkce nejsou nelineární interakce.
- Dostupné informace o gradientu.

**c) Stochastické**

Algoritmy založené na náhodě. Hodnoty argumentů účelové funkce jsou náhodně hledány a výsledek je vždy to nejlepší nalezené řešení celého náhodného hledání. Jsou pomalé a vhodné pro prohledávání velmi malý prostor možných řešení. Často vhodné pro hrubý odhad řešeného problému.

**d) Smíšené**

Algoritmy využívající výhod jak algoritmů deterministických, tak algoritmů stochastických. Smíšené algoritmy mají několik vlastností, s kterými dosahují velmi dobrých výsledků. Dokážou nalézt kvalitní řešení, dané jedním či více globálními extrémy, a to nezávisle na počátečních podmínkách a během relativně malého počtu ohodnocení účelové funkce. Požadavky na předběžné informace jsou minimální, příp. žádné a během jednoho spuštění jsou schopny nalézt i více řešení. Navíc jsou smíšené algoritmy schopny pracovat se systémy popisovanými též jako *černá skříňka*. Jde o problémy, u nichž neznáme popis účelové funkce.



Obr. 1-1. Rozdělení optimalizačních metod

### 1.3 Základní pojmy

*Účelová funkce* – funkce, matematický model problému, který se má optimalizovat. Označení účelové funkce je  $f(x)$ . Občas se lze setkat s názvem *cenová funkce*, pak je označována jako  $f_{\text{cost}}(x)$  podle anglického slova *cost*, které v češtině znamená *cena*. Příkladem účelové funkce může být Schwefelova funkce a vypadá následovně:

$$f(x) = \sum_{i=1}^n -x_i \sin(\sqrt{|x_i|}) \quad (1.1)$$

*Optimalizace* – proces nalezení maxima či minima účelové funkce. Výsledkem optimalizace je pak nalezení optimálních hodnot argumentů optimalizované funkce.

*Hodnota účelové funkce* – ohodnocení účelové s nalezenými argumenty. Porovnání hodnot účelové funkce pro různé argumenty vede k optimalizaci funkce, tj. k nalezení maxima či minima ať už globálního, tak lokálního.

*Vhodnost* – též označováno jako *Fitness* je upravená hodnota účelové funkce, zpravidla do intervalu (0; 1).

*Globální (lokální) minimum (maximu)* – funkce  $f$  má v bodě  $x_0$  lokální minimum, resp. maximum, existuje-li okolí bodu  $x_0$  takové, pro které platí:

$$f(x) \geq f(x_0) \quad \text{resp.} \quad f(x) \leq f(x_0) \quad (1.2)$$

pro všechna  $x$  z tohoto okolí. Funkce  $f$  má v bodě  $x_0$  globální minimum, resp. maximum, existuje-li okolí bodu  $x_0$  takové, pro které platí:

$$f(x) > f(x_0) \quad \text{resp.} \quad f(x) < f(x_0) \quad (1.3)$$

pro všechna  $x$  z tohoto okolí, mimo bod  $x = x_0$ .

*Jedinec* – jedincem se rozumí vektor, který obsahuje tolik čísel, kolik argumentů je optimalizováno v účelové funkci, obecně označován písmenem  $N$  (někdy též  $D$ ). Každý jedinec je jeden z možných řešení daného problému.

*Populace* – populace je množina jedinců, počet jedinců v populaci je označován písmenem  $M$ . Populace může být reprezentována jako matice  $N \times M$ .

*Specimen* – vzorový jedinec, dle kterého se generuje prvopočáteční populace jedinců. Tento vzorový jedinec může nabývat hodnot reálných, celočíselných, diskretních, atd. Specimen může vypadat například následovně:

$$\text{Specimen} = \{\{\text{Integer}, \{\text{Lo}, \text{Hi}\}\}, \{\text{Real}, \{\text{Lo}, \text{Hi}\}\}, \{\text{Integer}, \{\text{Lo}, \text{Hi}\}\}\} \quad (1.4)$$

kde první index určuje typ hodnoty a druhý index je interval určující spodní hranici  $\text{Lo}$  a horní hranici  $\text{Hi}$ .

*Křížení* – proces, při kterém křížením rodičů vznikají noví jedinci, potomci.

*Mutace* – proces, při kterém dochází k mutaci potomků. Mutace se projevuje, jako náhodná změna některých parametrů jedince pomocí náhodného procesu.

Pozn.: je-li proces mutace 100%, dochází ke změně celého jedince, a tudíž se mění i vhodně zvolený algoritmus na pouhé náhodné prohledávání.

*Evoluční kolo, Migrace, Generace* – jeden optimalizační cyklus, během kterého dochází k nalezení optimální hodnot účelové funkce.

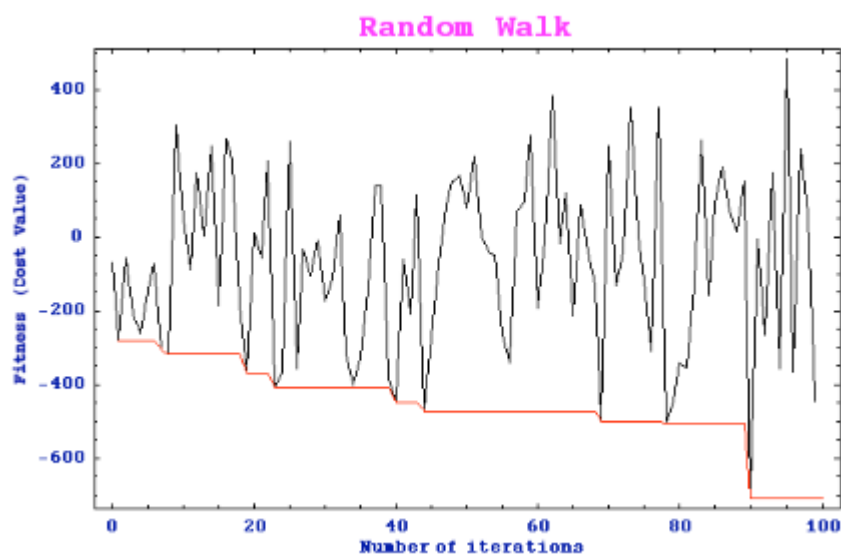
## 2 VYBRANÉ OPTIMALIZAČNÍ A EVOLUČNÍ TECHNIKY

### 2.1 Slepý algoritmus

Jedním z prvních optimalizačních algoritmů byl tzv. *Slepý algoritmus* někdy též nazývaný jako *algoritmus náhodné procházky* (*Random Walk*) (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009), či *náhodné prohledávání* (*Random Search*) (Tvrdík, 2004), který patří do skupiny algoritmů stochastických. Jde o algoritmus v celku jednoduchý a to i na naprogramování. Jediným parametrem je ukončovací parametr, určující počet optimalizačních kol, tzv. iterací.

Na začátku je vygenerován náhodný bod na dané účelové funkci a tento bod se uvažuje jako minimum, případně maximum. Následně algoritmus v každém kroku optimalizace náhodně generuje další bod na celé účelové funkci a tento bod je porovnán s minimem (maximem). Je-li nový bod menší (větší), uvažuje se za nové minimum (maximum). Počet vygenerovaných bodů odpovídá velikosti uživatelem zadané iterace.

Jak již z popisu vyplývá, algoritmus si pamatuje pouze aktuální hodnotu minima (maxima), ale nemá žádné vazby na kvality předchozích nalezených hodnot, nalezené hodnoty se mohou často opakovat a výsledné minimum (maximum) nemusí být právě globální extrém dané funkce. Typický běh slepého algoritmu lze vidět na obrázku (Obr. 2-1), převzato ze (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009).



Obr. 2-1. Příklad běhu slepého algoritmu s červeně naznačeným průběhem nejlepších dosažených řešení.

Obdobou slepého algoritmu (náhodného prohledávání) může být *řízené náhodné prohledávání (Controlled Random Search)* (Tvrdík 2004).

Pozn.: správné nalezení optimálních hodnot argumentů účelové funkce není problémem jen u algoritmu slepého, ale u každého optimalizačního algoritmu závisí správně nalezené optimum na vhodně nastavených parametrech daného algoritmu. Těmito parametry mohou být iterace (evoluční cykly, migrace, generace), velikost populace, ukončovací parametr MinDiv (*SOMA*), směrodatná odchylka pro Gaussovo normální rozdělení (*Evoluční strategie*) a mnoho dalších.

## 2.2 Horolezecký algoritmus

*Horolezecký algoritmus (Hill Climbing, HC)* patří mezi algoritmy deterministické, výjimkou je *Stochastický horolezecký algoritmus (Stochastic Hill Climbing)*, který využívá náhodného počátečního řešení  $x_0$  a tak se řadí mezi algoritmy stochastické.

Název horolezeckého algoritmu je odvozen od principu optimalizace, který připomíná horolezce (nejlepší hodnota každé iterace), který se šplhá do kopce (po dané funkci) až na vrchol hory (do extrému funkce). Ačkoli je horolezecký algoritmus popisován jako optimalizace, která hledá maximum dané funkce, tak se optimalizační algoritmy využívají především pro hledání minima funkce. Řešení tohoto problému je velice snadné, optimalizovanou funkci  $f$ , na které chceme hledat minimum v bodě  $x_0$ , vynásobíme číslem -1 a dostaneme funkci, na které nyní bod  $x_0$  znázorňuje maximum:

$$f_{max}(x) = f_{min}(x) * (-1) \quad (2.1)$$

A naopak optimalizovanou funkci  $f$ , na které chceme hledat maximum v bodě  $x_0$ , vynásobíme číslem -1 a dostaneme funkci, na které nyní bod  $x_0$  znázorňuje minimum:

$$f_{min}(x) = f_{max}(x) * (-1) \quad (2.2)$$

Pak můžeme na funkci, upravenou dle (2.1), aplikovat horolezecký algoritmus a nalezené maximum představuje minimum původní funkce.

Oproti předešlému algoritmu horolezecký algoritmus využívá postupu ve směru největšího gradientu. V každém kroku se negeneruje jeden bod na celé účelové funkci, ale množina bodů, tzv. populace a pouze v určitém okolí aktuálního nejlepšího řešení. Nalezené lokální maximum (minimum) tohoto okolí se pak stává „středem“ nového okolí. Při hledání lokálního maxima (minima) okolí se nebere v potaz střed okolí, z čehož vyplývá, že maximalizace (minimalizace) se opakuje vždy, a to i v případě, kdy nalezené lokální maximum (minimum) je horší než aktuální nejlepší řešení (střed okolí). Důsledkem toho pak může být tzv. *problém cyklických* řešení, kdy se algoritmus vrátí k řešení, které již bylo lokálním řešením v předchozím iteračním kroku.

Horolezecký algoritmus se řídí předpisem:

$$HC = (M, x_0, N, f, t_{max}), \quad (2.3)$$

kde:  $M$  – prostor řešení

$x_0$  – počáteční řešení  $x_0 \in M$

$N(x, \sigma)$  – množina sousedních řešení  $x \in M$

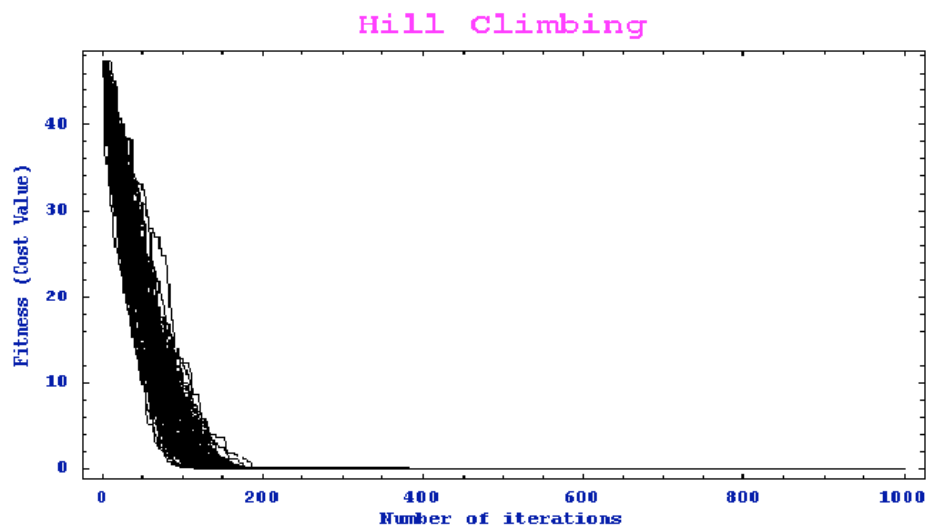
$f$  – optimalizovaná funkce

$t_{max}$  – zvolený počet iterací

Horolezeckého algoritmu může být několik variant:

- a) Jednoduchý horolezecký algoritmus (dané počáteční řešení  $x_0$ )
- b) Stochastický horolezecký algoritmus (náhodné počáteční řešení  $x_0$ )
- c) Horolezecký algoritmus s učením
- d) Paralelní horolezecký algoritmus

Typický běh horolezeckého algoritmu lze vidět na obrázku (Obr. 2-2), převzato ze (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009).



Obr. 2-2. Příklad běhu horolezeckého algoritmu pro unimodální funkci

Podrobněji o horolezeckém algoritmu a jeho modifikacích se lze dočíst v (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) nebo (Kvasnička, Pospíchal, Tiňo, 2000).

### 2.3 Simulované žihání

*Simulované žihání (Simulated Annealing, SA)* patří do algoritmů stochastických. Tak jako horolezecký algoritmus je inspirovaný horolezcem, který šplhá na horu, tak i algoritmus simulovaného žihání je inspirován fyzikálním procesem *žihání tuhého tělesa*.

Algoritmus simulovaného žihání je podobný horolezeckému algoritmu, připouští přijetí i horšího nalezeného řešení, než je aktuální nejlepší, ovšem oproti horolezeckému algoritmu je jedinec schopný překonat lokální extrém alespoň ze začátku optimalizace. Akceptace nového řešení optimalizace funkce je analogická přijetí nového stavu systému u žihání tuhého tělesa. Na začátku tuhnutí tělesa je teplota tělesa vysoká a nový stav je akceptován s vysokou pravděpodobností, jakmile teplota tělesa klesá, je pravděpodobnost přijetí nového stavu tělesa snižována, ke konci tuhnutí je pak blízká nule. Těleso již zůstává ve stejném tvaru. U optimalizace funkce je postup obdobný, dojde-li ke změně řešení optimalizace ze stavu  $x_0$  na  $x$ , podléhá přijetí nového řešení *Metropolisovu kritériu*, které je založeno na pravděpodobnosti  $P(x \rightarrow x_0)$  náhrady starého řešení takto:

$$P(x \rightarrow x_0) = \begin{cases} a) & 1, & \text{pro } f(x) < f(x_0) \\ b) & e^{-(f(x)-f(x_0))/T}, & \text{pro } f(x) \geq f(x_0) \end{cases} \quad (2.4)$$

Jestliže má nové řešení  $x$  menší funkční hodnotu, než původní řešení  $x_0$ , je nové řešení přijato s pravděpodobností 1 dle (2.4) a), pakliže má nové řešení  $x$  větší funkční hodnotu, je pravděpodobnost přijetí určena dle vztahu (2.4) b). Nové řešení je pak akceptováno jen tehdy, když  $P(x \rightarrow x_0) > r$ , kde  $r$  je náhodné číslo z intervalu  $(0, 1)$ .

Popsaný postup akceptace nového řešení je opakován pro jednu teplotu  $T$  tolikrát, jaké číslo obsahuje parametr  $n_T$ , který převážně nabývá hodnoty rovné počtu sousedů. Tímto opakováním je simulován proces žíhání tuhého tělesa a proces je nazýván jako *Metropolisův algoritmus*.

Algoritmus simulovaného žíhání se řídí předpisem:

$$SA = (M, x_0, N, f, T_0, T_f, \alpha, n_T), \quad (2.5)$$

kde:  $M, x_0, N(x, \sigma)$  a  $f$  jsou stejné jako u horolezeckého algoritmu.

$T_0$  – počáteční teplota

$T_f$  – konečná teplota, tzv. krystalizační

$\alpha$  – funkce redukce teploty. Každé teplotě přiřazuje teplotu o něco nižší:

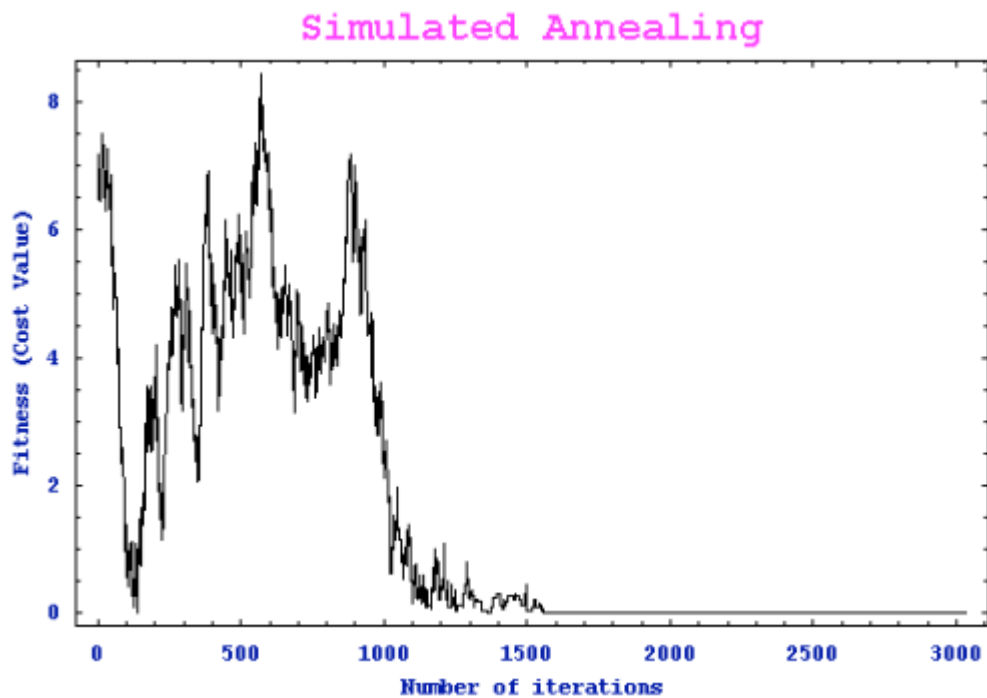
$$\alpha : T \rightarrow T', T' < T \quad (2.6)$$

$n_T$  – počet opakování *Metropolisova algoritmu* pro danou teplotu  $T$ . Převážně nabývá hodnoty rovné počtu sousedů.

Algoritmu simulovaného žíhání může být několik variant:

- a) Klasické simulované žíhání
- b) Simulované žíhání s elitismem
- c) Simulované žíhání paralelní

Typický běh algoritmu simulovaného žíhání lze vidět na obrázku (Obr. 2-3), převzato ze (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009).



Obr. 2-3. Příklad algoritmu simulovaného žíhání

Více o algoritmu simulovaného žíhání a jeho modifikacích se lze dočíst v (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) nebo (Kvasnička, Pospíchal, Tiňo, 2000).

## 2.4 Genetické algoritmy

*Genetické algoritmy (Genetic Algorithms, GA)* patří do algoritmů smíšených. Typickým rysem genetických algoritmů je jedinec, tzv. chromozóm, který je reprezentován binárním řetězcem. Každý parametr jedince je tedy v binární podobě a je nazýván gen.

Zatím co u výše popsáných algoritmů byla nová populace náhodně generována a závislost na rodiči (aktuální řešení, nejlepší řešení) byla pouze v jeho poloze, mimo slepého algoritmu, kde byla nová populace generována zcela náhodně, u algoritmů genetických se využívá principů používaných v přírodě, jež byly popsány pány Charlesem Robertem Darwinem a Gregorem Johannem Mendelem. Tak jako v přírodě i u GA dojde nejprve k výběru vhodných rodičů, nová populace vznikne křížením těchto rodičů a následně ještě nová populace podlehne mutaci.

### 2.4.1 Výběr rodičů

Dle Darwinovi teorie jsou pro vznik nové populace nejvhodnější jen nejlepší jedinci rodičů. Je zde ovšem místo pro úvahu, zda neustálým výběrem nejlepších rodičů nedojde k přešlechtění jedinců a místo, abychom získávali stále lepší jedince (výsledky optimalizace), dojde naopak k degeneraci populace. Tomuto by bylo možné zabránit občasným přijetím i horšího rodiče (řešení) ovšem s lepšími geny (parametry) pro vývoj nové populace.

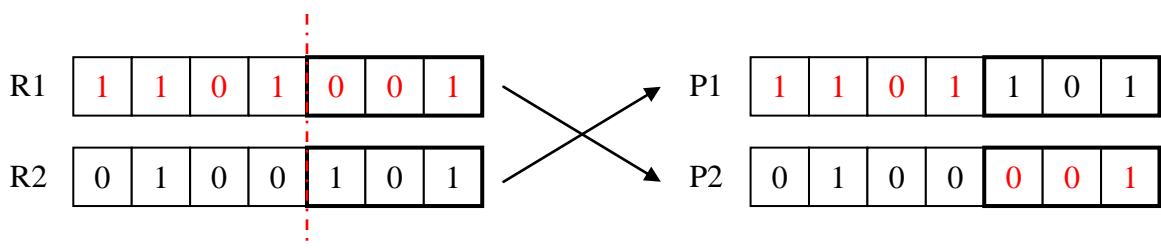
K výběru rodičů vhodných ke vzniku nové populace se dá využít dvou základních postupů, a to využití nepřerozdělených hodnot účelové funkce. Tato varianta je značně nevýhodná z důvodu velkých rozdílů hodnot účelové funkce mezi jedinci. Proto se využívá přerozdělených hodnot účelové funkce do intervalu  $(0, 1)$ , tzv. vhodnosti. Pro obě varianty je pak postup výběru rodičů následující: sečtou se hodnoty účelových funkcí, příp. vhodností, všech jedinců v populaci a označí se  $S$ . Následně se vygeneruje náhodné číslo  $r$  z intervalu  $(0, S)$ . Nakonec je populace procházena a počítá se suma  $s$  hodnot účelových funkcí, příp. vhodností, a jakmile hodnota  $s$  překročí hodnotu  $r$ , stává se jedinec rodičem pro další populaci.

Při výběru rodičů lze využít tzv. *elitismus*. Elitismus se využívá k tomu, aby nedocházelo ke ztrátě nelepších řešení, a to tak, že se z populace vybere nejlepší jedinec (nejlepší ohodnocení účelové funkce, příp. vhodnost), pak teprve následuje výběr dalších rodičů dle výše popsaného postupu.

### 2.4.2 Křížení a mutace

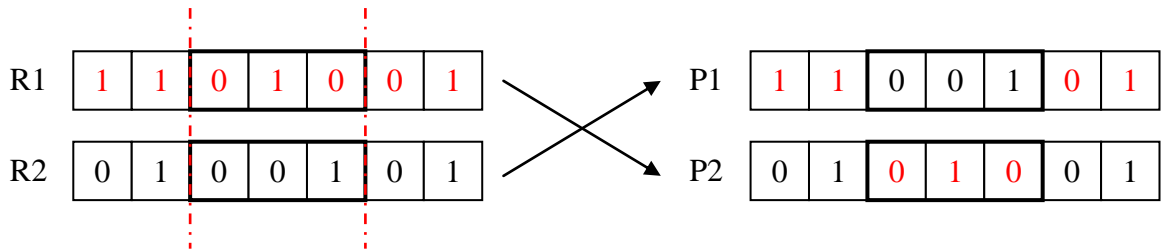
Po výběru rodičů nastává samotný proces generace nové populace. Toto je tvořeno dvěma kroky, a to *křížením* a *mutací*. Při procesu křížení jsou ze dvou rodičů tvořeni dva potomci, a to křížením (přehozením) části chromozómů rodičů. Křížení může být:

#### a) Jednobodové



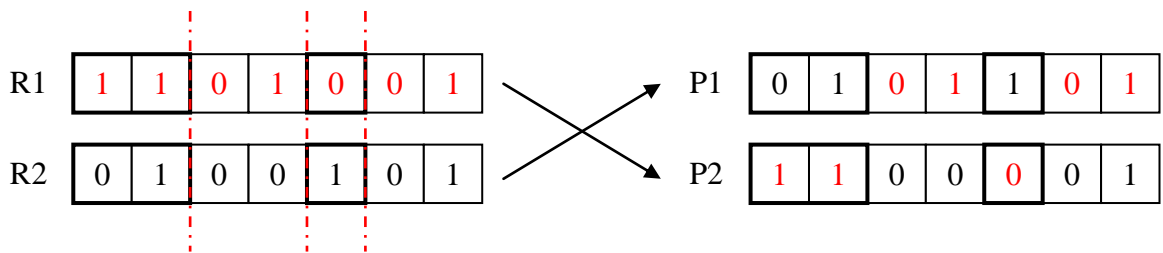
Obr. 2-4. Příklad jednobodového křížení

b) **Dvoubodové**



Obr. 2-5. Příklad dvoubodového křížení

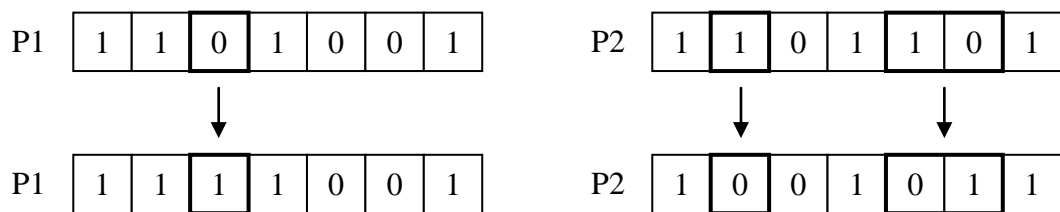
c) **Vícebodové**



Obr. 2-6. Příklad vícebodového křížení

Po procesu křížení přichází na řadu mutace každého jedince. Mutací rozumíme změnu určitých genů jedince. Mutace opět může být:

- a) **Jednobodová**
- b) **Dvoubodová**
- c) **Vícebodová**



Obr. 2-7. Příklad jednobodové a vícebodové mutace

Oba dva procesy, jak křížení, tak i mutace, se provádí dle pravděpodobnosti křížení, příp. mutace. Pravděpodobnost křížení  $P(K)$  a mutace  $P(M)$  splňují:

$$\frac{P(K)}{P(M)} = (0, 1) * 100\% \tag{2.7}$$

Pokud by byla pravděpodobnost křížení  $P(K) = 0\%$ , pak by nová populace potomků byla totožná s populací rodičů. Pokud by byla pravděpodobnost mutace  $P(M) = 100\%$ , pak by došlo ke změně všech genů potomka a genetický algoritmus se pak mění na pouhé náhodné prohledávání.

Genetických algoritmů může být několik variant:

- a) Klasický genetický algoritmus
- b) Hybridní genetický algoritmus
- c) Messy genetický algoritmus
- d) Paralelní genetické algoritmy

Podrobněji se genetickými algoritmy (klasický GA, hybridní GA, paralelní GA), ale i genetickým programováním zabývají (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009), (Kvasnička, Pospíchal, Tiňo, 2000) nebo Josef Hynek v (Hynek, 2008), který tomuto tématu věnuje celou publikaci. Ukazuje zde využití GA při problému N dam, problému obchodního cestujícího a dalších.

## 2.5 Evoluční strategie

*Evoluční strategie* (*Evolutionary Strategy*, ES) patří do algoritmů smíšených. Evoluční strategie navrhli pánové Schwefel a Rechenberg (Rechenberg, 1973) a byly vyvíjeny současně s genetickými algoritmy, ovšem oproti GA používaly ES jedince prezentovány v oboru reálných čísel. Další odlišností oproti GA je postup generování nové populace. Zatím co u GA populace potomků vznikla křížením rodičů a následnou mutací, u ES se místo křížení rodičů využívá *selekce* a následná mutace.

Pro označení ES se používá základní syntaxe:

$$(\mu, \lambda) - \text{ES}, \quad (2.8)$$

$$(\mu + \lambda) - \text{ES}, \quad (2.9)$$

- kde:  $\mu$  – označení pro populaci rodičů  
 $\lambda$  – označení pro populaci potomků  
 $,$  – symbol určující, že se do nové populace vybírají jen nejlepší řešení z populace potomků  
 $+$  – symbol určující, že se do nové populace vybírají nejlepší řešení jak z populace potomků, tak i z populace rodičů

a musí platit:

$$\mu \leq \lambda \quad (2.10)$$

Evoluční strategie může být několik variant:

- Dvoučlenné evoluční strategie:  $(1 + 1) - ES$
- Vícečlenné evoluční strategie:  $(\mu, \lambda) - ES$  a  $(\mu + \lambda) - ES$
- Rekombinační evoluční strategie:  $(\mu/\rho + \lambda) - ES$
- Adaptivní evoluční strategie
- Sebeadaptivní evoluční strategie

### 2.5.1 Dvoučlenné evoluční strategie

Syntaxe dvoučlenné ES je dle vztahu (2.9) a vypadá takto:  $(1 + 1) - ES$ . Dvoučlenné ES jsou nejjednodušší verze ES, pracují s jedním rodičem a jedním potomkem, který je určen pomocí tzv. *Gaussova mutačního operátoru* a řídí se předpisem:

$$ES = (\mu, \sigma, f_{cost}, iterace, FV), \quad (2.11)$$

- kde:  $\mu$  – prvopočáteční náhodně vygenerovaný rodič  $x_i$   
 $\sigma$  – směrodatná odchylka pro Gaussovo normální rozdělení  
 $f_{cost}$  – účelová funkce vracející vhodnost  
 $itera$  – maximální počet iterací  
 $FV$  – vhodnost, ukončovací parametr. Při dosažení vhodnosti se ES ukončí.

### 2.5.2 Vícečlenné evoluční strategie

Syntaxe vícečlenné ES je dle vztahu (2.8) i (2.9). Pracují s  $\mu$  rodiči a  $\lambda$  potomky a řídí se stejným předpisem jako dvoučlenné ES.

S vícečlennými ES se zavádí parametr  $\kappa \geq 1$  tzv. *životnost*, který udává maximální stáří rodiče, neboli, kolikrát smí být rodič vybrán do další populace. Pokud stáří rodiče překročí  $\kappa$ , není již vybrán do další populace.

### 2.5.3 Rekombinační evoluční strategie

Syntaxe rekombinační ES je dle vztahu (2.9) doplněná o parametr  $\rho \in (1, \mu)$  a vypadá takto:

$$(\mu/\rho + \lambda) - \text{ES} \quad (2.12)$$

Parametr  $\rho$  určuje, z kolika rodičů se bude počítat tzv. *rekombinant*. Rekombinace je analogická k procesu křížení u GA a mutací vzniklého rekombinantu pak vzniká nový potomek. Rekombinace může být:

a) **Průměrová**

$$y_{rek} = \frac{1}{\rho} \sum_{i=1}^{\rho} x^i \quad (2.13)$$

b) **Diskrétní**

Každý parametr rekombinanta je náhodně vybrán z  $\rho$  rodičů.

Při použití ES bylo definováno pravidlo 1/5:

*„Poměr ps úspěšných mutací ku všem mutacím by měl být 1/5. Pokud je aktuální poměr mutací větší, pak se musí zvýšit intenzita mutace, pokud je aktuální poměr mutací menší, pak se musí zmenšit intenzita mutace.“ (Rechenberg, 1973)*

Více se evolučními strategiemi zabývají (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009), (Kvasnička, Pospíchal, Tiňo, 2000) nebo (Tvrdík, 2004).

## 2.6 ACO

*Optimalizace mravenčí kolonie (Ant Colony Optimization, ACO)* (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) patří do algoritmů smíšených. ACO je další z algoritmů, které jsou inspirovány přírodou, přesněji kolonií mravenců hledajících nejlepší cestu za potravou.

V přírodě mravenci prohledávají okolí mraveniště a hledají potravu. Jakmile mravenec potravu najde, vrací se do mraveniště a zanechává za sebou feromonovou stopu. V momentě, kdy na ni jiný mravenec narazí, vydá se po stopě k potravě. Po cestě do mraveniště také za sebou zanechává feromon. Tímto se cesta stává intenzivnější a tudíž dominantnější. Cílem mravenců je nalézt maximum potravy s minimálně urazenou cestou. Představit si to lze na příkladu, kdy se do cesty mravencům postaví překážka. Jelikož nemohou dále pokračovat po stopě feromonu, začnou překážku obcházet v obou směrech, a to proto, že okolí překážky doposud nebylo prozkoumáno a tudíž neobsahuje žádné feromonové stopy. Po chvíli ovšem začne kratší cesta obsahovat více feromonu, proto začnou mravenci chodit právě touto kratší cestou, na delší cestě se začne feromon vytrácet a mravenci nosí více potravy za minimální dobu.

Pohyb  $k$ -tého mravence mezi dvěma body lze popsat pravděpodobností:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \quad (2.14)$$

kde:  $\tau_{ij}$  – feromonová stopa mezi body  $i$  a  $j$  (dále dle (2.15))

$\alpha$  – parametr kontroly vlivu na  $\tau_{ij}$

$\eta_{ij}$  – požadavek na cestu mezi body  $i$  a  $j$

$\beta$  – parametr kontroly vlivu na  $\eta_{ij}$

$$\tau_{ij} = \rho \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k, \quad (2.15)$$

kde:  $\rho$  – poměr vypařování feromonu

$m$  – počet mravenců

$\Delta \tau_{ij}^k$  – množství naneseného feromonu  $k$ -tým mravencem mezi body  $i$  a  $j$  (dále dle (2.16))

$$\Delta\tau_{ij}^k = \begin{cases} a) & Q/C^k, & \text{jestliže se mravenec } k \text{ pohybuje mezi body } i \text{ a } j, \\ b) & 0, & \text{jinak} \end{cases} \quad (2.16)$$

kde:  $Q$  – množství feromonu uložené mravencem

$C^k$  – ohodnocení cesty  $k$ -tého mravence

Optimalizace mravenčí kolonií může být několik variant:

- a) Klasická optimalizace mravenčí kolonií
- b) Mravenčí systém s elitismem
- c) Max-Min mravenčí systém
- d) Rank-Based mravenčí systém

## 2.7 SOMA

*SamoOrganizující se Migrační Algoritmus* (SOMA) spadá do algoritmů smíšených. SOMA algoritmus, jehož tvůrcem je Ivan Zelinka (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009), se tak jako předešlé algoritmy také řídí zákony v přírodě, ovšem tentokrát se nejedná o evoluční zákony jako takové, ačkoli tento algoritmus spadá do algoritmů evolučních, nýbrž o zákony vycházející, jak již samotný název napovídá, ze spolupráce populace migrující po prostoru.

Oproti předešlým algoritmům SOMA nevyužívá procesy *křížení* a *mutace* v pravém slova smyslu. I parametr označovaný jako *generace* je zde nahrazen parametrem *migrace*, či *migrační kolo*. Obecně si lze průběh algoritmu představit, jako populaci jedinců, která putuje po prostoru a hledá globální minimum (maximum) účelové funkce, analogický přirovnáváno k přírodě, např. potravu, ložisko ropy, zlata, nejkvalitnější půdu apod. Každý jedinec se ohodnotí, zjistí aktuální kvalitu půdy, stav potravy v okolí apod. a sdělí si mezi sebou kvalitu řešení, převážně chvástáním. Ostatní jedinci se pak pohybují, „kříží“ a „mutují“, ve směru nejlepšího jedince, řešení. Při pohybu k nejlepšímu jedinci opět dochází k ohodnocení a sdílení kvality řešení, mezi jedinci.

Prvotní populace je dána dle *Specimena* (viz, kapitola 1.3 Základní pojmy).

### 2.7.1 Křížení a mutace

Křížením zde není chápáno jako generování nové populace ze staré, ale generování posloupnosti  $t$  potomků (2.17) jednoho jedince po diskretních krocích ve směru k nejlepšímu jedinci. Z této posloupnosti pak přežívá pouze nejlépe ohodnocený potomek, čímž je určena nová pozice jedince.

$$t = \frac{\text{PathLength}}{\text{Step}}, \quad (2.17)$$

kde: PathLength – délka cesty jedince, jak daleko se jedinec pohybuje před, za vedoucího jedince.

Step – krok jedince, po jaké vzdálenosti se budou generovat potomci jedince.

Mutací opět není chápána změna parametrů jedince, ale náhodná změna směru každého potomka při procesu křížení, tento proces je nazýván *perturbace* a jedinec tak nepodléhá mutaci, nýbrž *perturbaci*. Vliv *perturbace* na jedince je dán náhodně generovaným číslem  $rnd \in (0, 1)$  a parametrem  $PRT \in (0, 1)$ , dle kterého se tvoří *perturbační vektor* PRTVektor:

$$\text{PRTVektor}_j = \begin{cases} a) & 1, \text{ jestliže } rnd_j < PRT \\ b) & 0, \text{ jestliže } rnd_j > PRT \end{cases}, \text{ pro } j = 1, \dots, t, \quad (2.18)$$

kde: PRTVektor<sub>j</sub> – j-tý prvek *perturbačního vektoru* pro j-tého potomka jedince.

$rnd_j$  – náhodně generované číslo z intervalu  $(0, 1)$  pro určení *perturbačního vektoru* j-tého jedince.

Pohyb jedince je potom dán směrovým vektorem:

$$\vec{r} = \vec{r}_0 + \vec{m} t \overrightarrow{\text{PRTVektor}}, \text{ pro } t \in [0, \text{Step}, \text{PathLength}], \quad (2.19)$$

kde:  $r$  – odpovídá  $x_{i,j}^{M+1}$  a značí j-tý prvek i-tého jedince v migračním kole  $M+1$ .

$r_0$  – odpovídá  $x_{i,j,start}^M$  a značí startovní pozici j-tého prvku i-tého jedince v migračním kole  $M$ .

$m$  – značí rozdíl mezi pozicí vedoucího jedince  $x_{L,j}^M$  a startovní pozicí j-tého prvku i-tého jedince v migračním kole  $M$  a je dán vtahem:

$$m = x_{L,j}^M - x_{i,j,start}^M \quad (2.20)$$

Vstupními parametry algoritmu SOMA jsou:

PathLength, Step, PRT – popsány výše

$D$  – počet argumentů účelové funkce (počet optimalizovaných proměnných).

PopSize – řídicí parametr, velikost populace (kolik jedinců bude populaci tvořit).

Migrace – ukončovací parametr, počet migračních kol, podobně jako generace.

MinDiv – Minimal Diversity, ukončovací parametr, určuje maximální povolený rozdíl, mezi nejlepším a nejhorším jedincem populace v aktuální migraci.

SOMA algoritmu může být několik variant:

- a) SOMA All-To-One (všichni k jednomu)
- b) SOMA All-To-One-Rand (všichni k jednomu náhodně)
- c) SOMA All-To-All (všichni ke všem)
- d) SOMA All-To-All-Adaptive (všichni ke všem adaptivně)

Dále se o algoritmu SOMA lze dočíst v (Tvrdík, 2004).

## 2.8 Diferenciální evoluce

*Diferenciální evoluce* (DE) také spadá do algoritmů smíšených. Zajímavostí u DE je skutečnost, že tento algoritmus nevznikl jako nový algoritmus EVT, ale vznikl na základě *genetického žihání*. Úpravou genetického žihání, konkrétně změnou reprezentace jedince z binárního na dekadického a logických operací na vektorové, přidáním *diferenciální mutace* a paradoxně vypuštěním principů genetického žihání, jež se ve výsledku ukázaly nadbytečné, vznikly diferenciální evoluce takové, jaké je známe dnes.

Vstupními parametry diferenciálních evolucí jsou:

- CR – řídicí parametr  $\in (0, 1)$ , práh křížení.
- D – počet argumentů účelové funkce (počet optimalizovaných proměnných).
- NP – řídicí parametr, velikost populace (kolik jedinců bude populaci tvořit), minimálně 4 jedinci (viz. generování nové populace).
- F – řídicí parametr  $\in (0, 2)$ , mutační konstanta.
- Generations – ukončovací parametr, počet generací.

Běh DE je zahájen, jako u většiny evolučních algoritmů, vytvořením prvotní populace, která je dána dle *Specimena* (viz, kapitola 1.3 Základní pojmy). DE běží ve dvou cyklech, z čehož vnější cyklus je dán počtem generací (*Generations*) a vnitřní cyklus je dán počtem jedinců v populaci (*NP*). Z populace jsou pak vybíráni rodiče pro generování nové populace potomků.

### 2.8.1 Mutace a křížení

DE podobně jako předchozí algoritmy také využívají křížení a mutace jedinců k vytvoření nové populace potomků. Rozdílem oproti ostatním algoritmům ovšem je to, že tyto dva procesy probíhají v opačném pořadí, než jsme tomu byly zvyklí doposud. Další odlišností je počet vybraných rodičů pro generování nové populace, a to čtyř místo dvou.

Před samotným popisem postupu je vhodné ustanovit několik pojmů:

- **Vektor** – jedinec, parametry jedince.
- **Diferenční vektor** – vektor daný rozdílem dvou rodičů  $r_1$  a  $r_2$ .
- **Váhový diferenční vektor** – diferenční vektor vynásobený mutační konstantou  $F$ .
- **Šumový vektor** – součet váhového diferenčního vektoru a třetího rodiče  $r_3$ .
- **Cílový vektor** – parametry rodiče  $r_4$ .
- **Zkušební vektor** – výsledek složení z šumového a cílového vektoru pomocí konstanty prahu křížení  $CR$ .

Jak již bylo výše popsáno, DE běží ve dvou cyklech. Vnitřní cyklus je dán počtem jedinců v populaci, neboli, proběhne právě tolikrát, kolik je jedinců v populaci ( $NP$ ). V každém kroku cyklu je vybrán jeden jedinec jako rodič  $r_4$ , tzv. *cílový vektor*, pokaždé jiný, aby se všichni jedinci vystřídali, a k němu jsou náhodně vybráni další tři jedinci jako rodičové  $r_1$ ,  $r_2$  a  $r_3$ , obecně *vektory*, a z těchto čtyř rodičů je generován právě jeden potomek. Odečtením rodičů  $r_1$  a  $r_2$ , resp. jednotlivých složek vektorů, vznikne nový vektor, tzv. *diferenční*. Nyní přichází na řadu mutace, a to tak, že vzniklý diferenční vektor se vynásobí mutační konstantou  $F$  a tímto vznikne nový vektor, tzv. *váhový diferenční*. Tento vektor se sečte s vektorem rodiče  $r_3$  a vznikne *šumový vektor*, označovaný  $v_s$ :

$$v_{s,j} = x_{r_{3,j}}^G + F(x_{r_{1,j}}^G - x_{r_{2,j}}^G), \text{ pro } j = 1, \dots, D, \quad (2.21)$$

kde:  $v_{s,j}$  –  $j$ -tý prvek šumového vektoru.

$x_{r_{3,j}}^G$  –  $j$ -tý prvek vektoru rodiče  $r_3$  v generaci  $G$ . Stejně je to pro  $x_{r_{1,j}}^G$  a  $x_{r_{2,j}}^G$ .

Teprve nyní se provádí proces křížení za pomoci *cílového vektoru* (rodiče  $r_4$ ) a parametru křížení  $CR$ , výsledkem křížení je pak *zkušební vektor*, označovaný  $v_z$ :

$$v_{z,j} = \begin{cases} a) & v_{s,j}, \text{ jestliže } rnd < CR \\ b) & x_{r_{4,j}}^G, \text{ jestliže } rnd > CR \end{cases}, \text{ pro } j = 1, \dots, D, \quad (2.22)$$

kde:  $v_{z,j}$  –  $j$ -tý prvek zkušební vektoru.

$x_{r_{4,j}}^G$  –  $j$ -tý prvek cílového vektoru (rodiče  $r_4$ ) v generaci  $G$ .

$rnd$  – náhodně generované číslo z intervalu  $(0, 1)$ .

Vztah (2.22) lze slovně popsat takto: pro každý prvek zkušební vektoru  $v_z$  je vygenerováno náhodné číslo  $rnd$  z intervalu  $(0, 1)$ . Je-li toto náhodné číslo  $rnd$  menší než parametr křížení  $CR$ , je do zkušební vektoru na danou pozici umístěn prvek ze šumového vektoru. Je-li toto náhodné číslo  $rnd$  větší než parametr křížení  $CR$ , je do zkušební vektoru na danou pozici umístěn prvek z cílového vektoru. Takto vzniklý zkušební vektor se ohodnotí přes danou účelovou funkci a výsledné ohodnocení se porovnává s ohodnocením cílového vektoru (s ohodnocením rodiče  $r_4$ ) a do nové populace jako potomek postupuje ten z nich, jehož ohodnocení je menší.

Diferenciální evoluce může být několik variant, které se liší postupem vzniku šumového vektoru:

- a) Výše popsaná DE s náhodně vybraným rodičem  $r_3$  v každém
- b) DE s výběrem nejlepšího jedince v každém kole namísto rodiče  $r_3$
- c) a další

Dále se o diferenciální evoluci lze dočíst v (Tvrdík, 2004) nebo (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009).

## 2.9 No Free Lunch Teorém

No Free Lunch Teorém, jak už název napovídá, není optimalizační algoritmus, ale teorém, který říká, že:

- Neexistuje algoritmus, který by dokázal řešit všechny problémy lépe, než všechny ostatní algoritmy, nebo-li
- Existuje skupina problémů, pro které je algoritmus X lepší, než algoritmus Y, a také naopak existuje skupina problémů, pro které je lepší algoritmus Y, než algoritmus X, nebo-li
- Neexistuje „Bůh“ mezi algoritmy, který by stačil na všechny problémy.

V opačném případě by nemuselo existovat toliko algoritmů a stačil by jen jeden jediný algoritmus na všechny problémy.

### 3 IDENTIFIKACE PROMĚNNÝCH HVĚZD

Hvězdná obloha a hvězdy samotné přitahují pohledy lidí už od pradávna. Z postavení hvězd a jejich jasností lidé četli horoskopy, přicházející bídu či plodné období, nemoci, budoucnost, sny a mnohé další.

#### 3.1 Proměnné hvězdy

Proměnné hvězdy a soustavy hvězd se dělí do dvou základních skupin a tříd, jak je vidět na obrázku (Obr. 3-7):

##### 1. Geometrické proměnné hvězdy a soustavy

Jsou takové, u kterých se nemění světelný tok (jas), ale pouze jejich pozorovaná svítivost. Jinak řečeno, ke změnám jasu dochází díky rotaci hvězdy, příp. díky pohybu soustavy kolem společného těžiště.

##### 2. Fyzické proměnné hvězdy a soustavy

Jsou takové, u nichž opravdu dochází ke změnám zářivého výkonu, a to ať už v okolí hvězdy (soustavy), na jejím povrchu či v jádru hvězdy.

#### 3.1.1 Geometrické proměnné hvězdy

Geometrické proměnné hvězdy se dále dělí do dvou tříd:

##### 1. Rotující

Změna jasu je dána osou rotace, která nesměruje k pozorovateli. Svůj vliv na toto má i magnetické pole, jehož osa rotace je odlišná od osy rotace hvězdy.

##### 2. Dvojhvězdy

Jak již z názvu vyplývá, jde o dvě hvězdy, označované jako složky. Tyto složky se při své rotaci vzájemně vůči pozorovateli zakrývají. V těsných dvojhvězdách se uplatňuje i efekt odrazu, což má za následek vzájemné osvětlování složek. Dochází tak k rozptýlení světla ve fotosféře druhé složky, tudíž i k jeho absorbování a následného ohřevu fotosféry, což vede

ke zvýšení jasů. Při p̑etoku hmoty mezi složkami lze pozorovat tzv. *zákryty akrečním diskem*, příp. *plynnými proudy*, viz. následující obrázek.



Obr. 3-1. Akreční disk při p̑etoku hmoty dvojhvězdy  
(<http://www.hvezdnouoblohou.wz.cz/velryba.php>)

Větší hvězda obsahuje heliové jádro a bývá nazývána jako *obr*, někdy též *obr červený*. Menší, bílá hvězda je tvořena heliem a vzniká odhalením degenerovaného jádra obra a bývá nazývána jako *bílý trpaslík (BT)*.

### 3.1.2 Fyzické proměnné hvězdy

Geometrické proměnné hvězdy se dále dělí do tříd:

#### 1. Nestacionární děje

- a) V okolí hvězdy – dáno materiálem okolo hvězdy, jako jsou mlhoviny nebo odvržená obálka po výbuchu novy, příp. supernovy. V dvojhvězdě pak akrečním diskem, ve kterém při turbulentním pohybu klesá látka směrem k menší složce (BT) a tím dochází ke zjasnění hvězdy.
- b) Na povrchu hvězdy – při dopadu hmoty na povrch hvězdy dochází ke stlačování hvězdy, až následně dojde k explozi povrchových vrstev hvězdy a tím ke zjasnění o 7 – 19 magnitud. Typický děj pro novy s frekvencí opakování děje  $10^5$  let.

## 2. Povrchová aktivita hvězdy

Dáno chladnými skvrnami na povrchu hvězdy, podobně jako u Slunce. Svůj vliv na toto mají, převážně lokální, magnetická pole. Často doprovázeno erupcemi silnějšími než na Slunci

## 3. Pulzující proměnné hvězdy

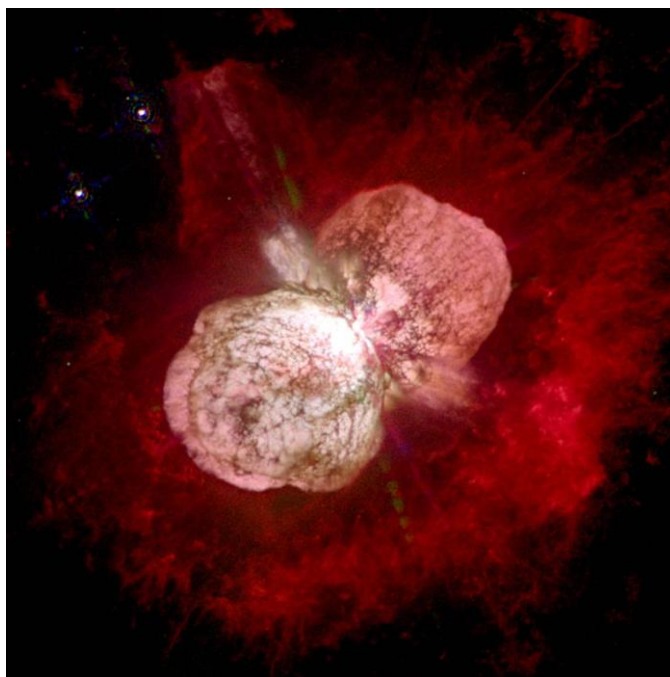
- a) Radiální pulzace – v hydrostatické rovnováze je zde gravitační síla s gradientem tlaku. Perioda pulzací je úměrná periodě vlastních kmitů. Nejvíce významnou je oblast sousedících vrstev jedenkrát ionizovaných atomů helia, označovanou HeII a zcela ionizovaných atomů helia, označovanou HeIII. Stlačováním HeII dochází k *ionizaci* na HeIII, následkem je ztmavnutí hvězdy. Zjasnění (zprůhlednění) hvězdy je pak následnou *expanzí* a *rekombinací* HeIII na HeII. Tento proces se neustále opakuje.
- b) Neradiální pulzace – hvězdu chápeme jako *prostorový rezonátor*, podobně jako Zemi při zemětřesení. Vlny se ve fotosféře odráží zpět ke hvězdě. Při pohybu zpět roste hustota a vlna se láme a odchyluje od kolmice, až opět dosáhne fotosféry. Zde se odráží a interferuje sama sebe, vzniká *stojaté vlnění*.
- c) Dlouhoperiodické proměnné – podobné, jako u radiální pulzace.

## 4. Supernovy

Označení pro exploze „ukončení života“ hvězd následkem chemické změny jádra hvězdy.

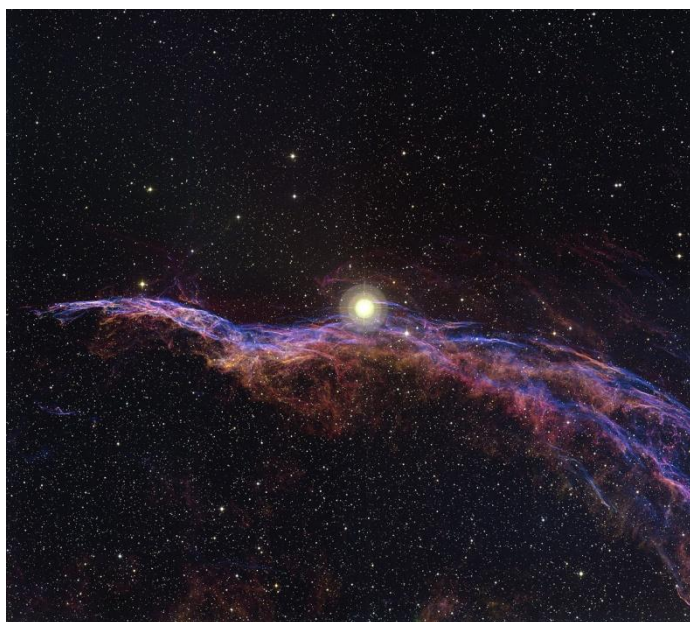
- a) Typu II – supernova (exploze) je výsledkem změny He jádra hvězdy na Fe. Dochází k porušení hydrostatické rovnováhy, v jádru roste hustota látky až do kritické hustoty. Uvolní se obrovské množství energie formou *neutrin*, látka se ohřívá, vzniká mohutná *rázová vlna* a dochází k explozi.
- b) Typu Ia – jedná se o bílého trpaslíka (menší složka dvojhvězdy), která přijímá „krade“ hmotu od obra (větší složka), viz. obázek (Obr. 3-1). V momentě, kdy hmotnost hvězdy překročí kritický limit 1,3 – 1,4 násobek hmotnosti Slunce, hvězda exploduje.
- c) Typu Ib a Ic – podobné, jako typu II.

Pohled na supernovu je prostě fascinující, z tohoto důvodu je zde přiloženo pár obrázků supernov.



Obr. 3-2. Supernova (<http://cs.wikipedia.org/wiki/Soubor:Supernova.jpg>)

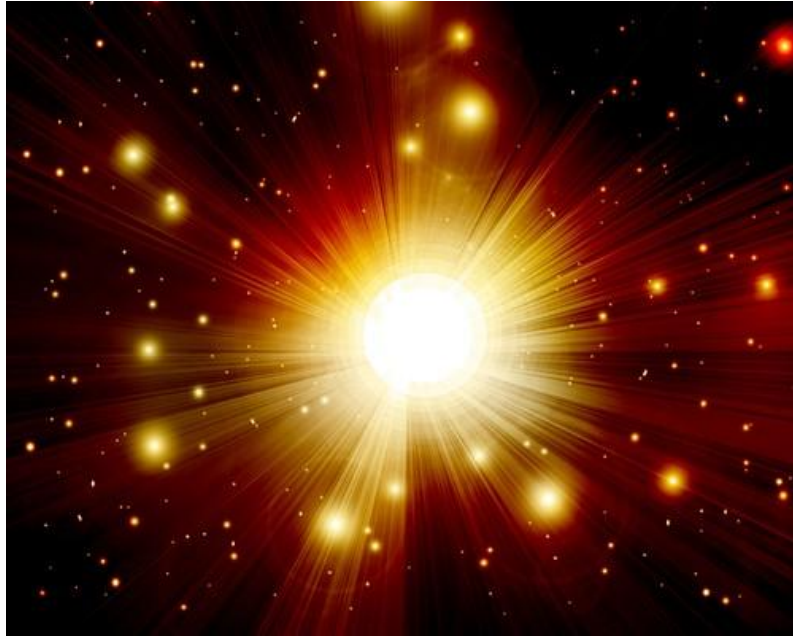
Supernova NGC 6960, nazývána jako *Košť čarodějnice* se nachází se v souhvězdí Labutě (lat. Cygnus).



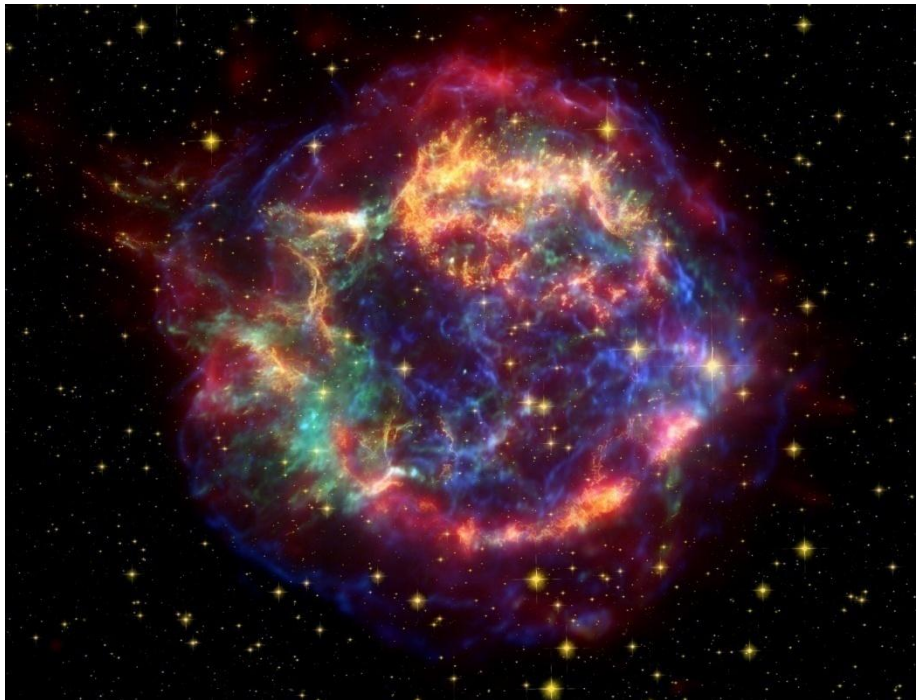
Obr. 3-3. Zbytky supernovy (NGC 6960)

(<http://objekty.astro.cz/mlhoviny/2208-stadia-zbytku-supernov>)

Supernova v galaxii NGC 1821 v souhvězdí Zajíce (lat. Lepus). Dle dlouhého bádání jde o dva bíle trpaslíky, kdy jeden okrádal druhého, až explodoval.

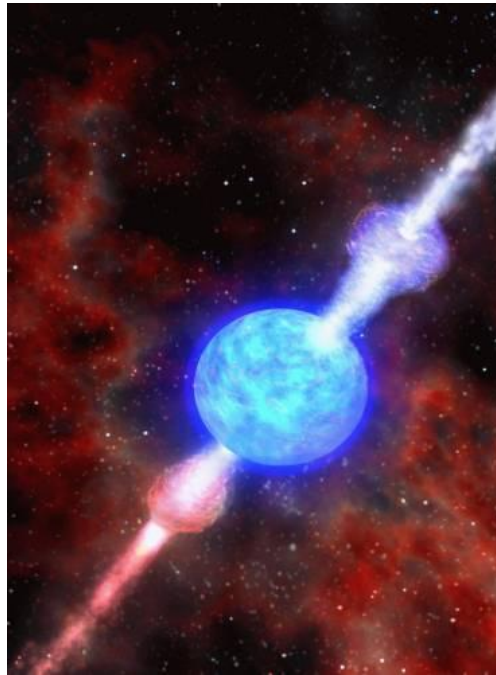


*Obr. 3-4. Supernova pravděpodobně typu Ia (<http://vtm.zive.cz/aktuality/loupezivy-bily-trpaslik-explodoval>)*

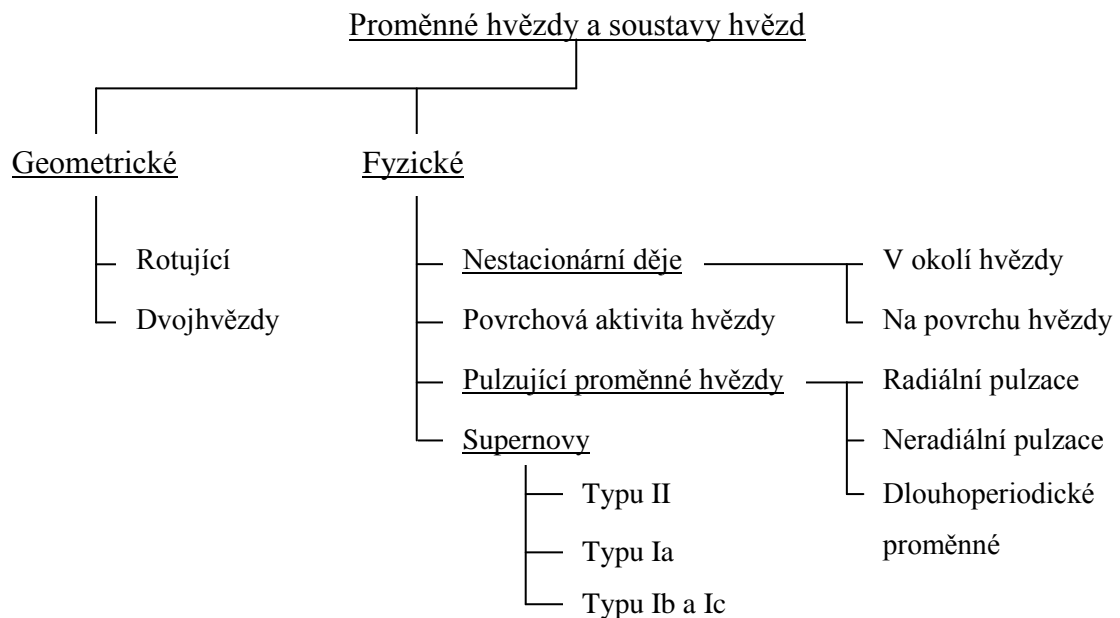


*Obr. 3-5. Pozůstatek supernovy, jejíž světlo roku 1680 zasáhlo i Zemi ([http://www.mpa-garching.mpg.de/mpa/institute/news\\_archives/news1005\\_janka/news1005\\_janka-de.html](http://www.mpa-garching.mpg.de/mpa/institute/news_archives/news1005_janka/news1005_janka-de.html))*

Supernova SN2006aj ležící v souhvězdí Berana (lat. Aries).



Obr. 3-6. Supernova SN2006aj (<http://21stoleti.cz/blog/2006/07/22/poprvе-ampquotv-primem-prenosuampquot-v-blizke-galaxii-explodovala-supernova/>)



Obr. 3-7. Rozdělení proměnných hvězd a soustav hvězd

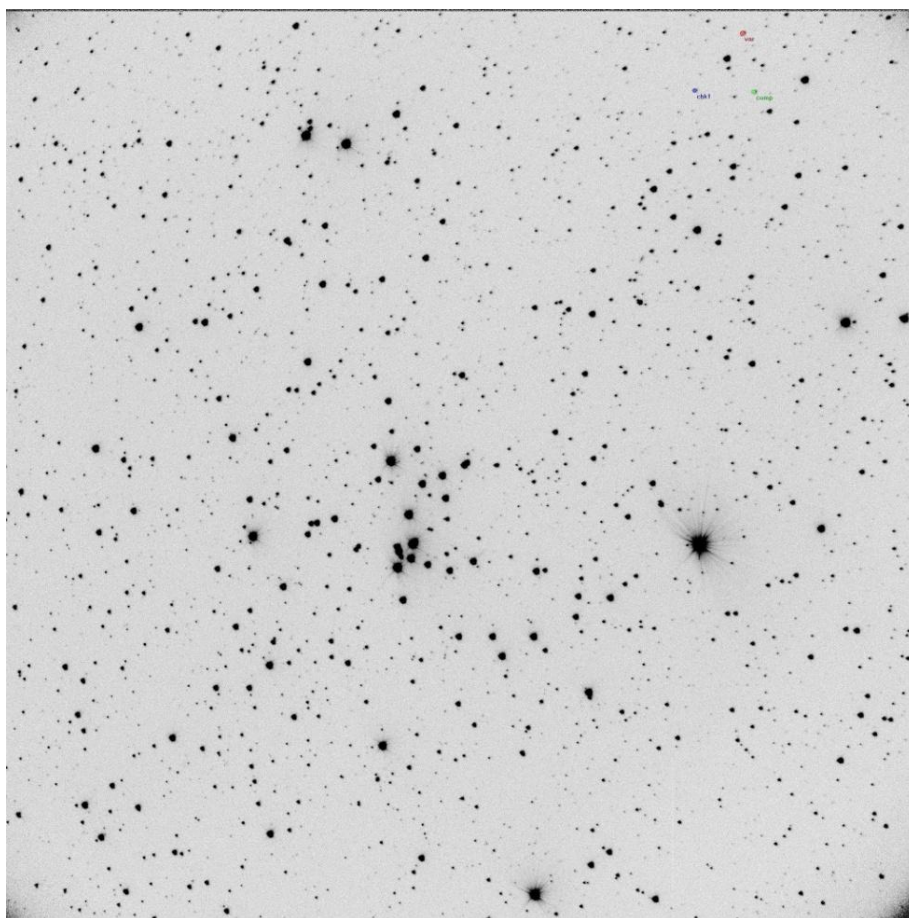
Podrobněji se proměnnými hvězdami a jejich pozorováním zabývají (Brát, Šmelcer a Trnka, 2011).

### 3.2 Pozorování a měření hvězd

Pro běžné pozorování hvězdné oblohy nám stačí naše oči, jimi jsme schopni rozpoznat základní hvězdy, jako jsou Polárka (Severka, Jitřenka, Večerka), Vega, Capella, Sírius a souhvězdí, jako Orión, Velký a Malý vůz, resp. Velkou a Malou medvědic, Kasiopeu, Pegas a mnoho dalších. Ovšem pro bližší pozorování a zaměřování polohy nám toto nestačí a přichází na řadu pomůcky od klasického dalekohledu (triedr), přes puškohledy (optisan) až po speciální hvězdářské dalekohledy a teleskopy. Mezi nejznámější hvězdářské dalekohledy patří Keplerův dalekohled, Galileův dalekohled, Newtonův dalekohled, Cassegrainův dalekohled, nebo Hubbleův vesmírný dalekohled (*HST, Hubble Space Telescope*), který obíhá Zemi po její oběžné dráze ve výšce 600 km již od roku 1990 a svojí polohou tak získává velmi ostré snímky vesmírných těles, které nejsou ovlivněny zemskou atmosférou.

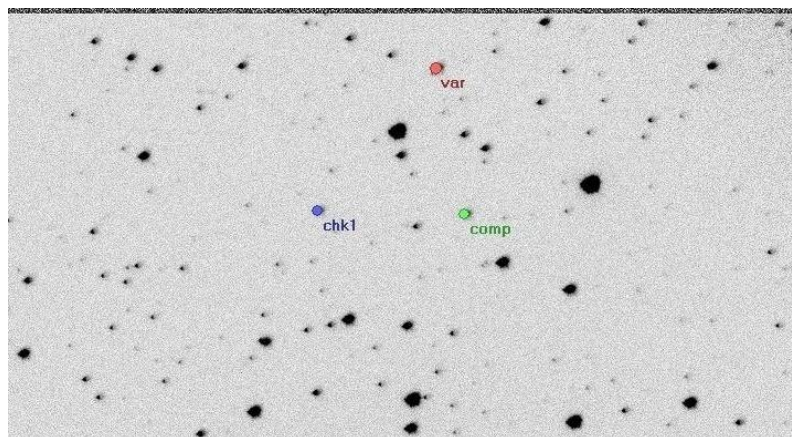
Časem potřeby pozorovatelů vzrostly, k čemuž dopomohly *CCD kamery*, které v roce 1969 vynalezli pánové Willard Boyle a George E. Smith. *CCD kamery (Charge-Coupled Device)* pracují na principu *fotoefektu*. Těchto kamer je nepřeberné množství druhů a typů, využívají se v široké škále odvětví, umožňují propojení s PC pomocí USB kabelu a díky své ceně jsou dostupné i pro amatérské hvězdáře. Jsou schopny nejen zaměřit polohu hvězdy, ale také uložit obrázek snímaného kousku oblohy, změřit jas hvězdy a další.

Pro vyhodnocování dat naměřených *CCD kamerami* slouží mnoho softwarových produktů, které jsou více či méně dobré, ovšem rozhodující pro uživatele je jejich cena. Proto mezi většinou hvězdářů vítězí softwarový balík *C-Munipack* od autorů Ing. Davida Motla a Filipa Hrocha, který je právě zdarma a je volně stažitelný z internetových stránek projektu: <http://c-munipack.sourceforge.net/>. *C-Munipack* je balík programů, které slouží pro práci se snímky pořízených právě *CCD kamerami* a jsou zaměřeny na fotometrii proměnných hvězd. Podrobněji se o projektu *C-Munipack* lze dočíst na stránkách projektu: <http://c-munipack.sourceforge.net/>, případně v uživatelském manuálu (Motl, 2008). Pořízené snímky lze programem zobrazit, editovat a uložit. Příklad mapky je na obrázku (Obr. 3-8). Naměřená data lze dále zpracovávat a ukládat do souboru, a to s příponou *SRT*, je-li soubor výstupem fotometrie, nebo s příponou *MAT*, jsou-li soubory výsledkem skládání. V praktické části této práce se budeme dále zabývat soubory *MAT*, příklad souboru lze vidět na obrázku (Obr. 4-2).



*Obr. 3-8. Otevřená hvězdokupa NGC 2281 v programu C-Munipack*

Jak lze vidět v pravém horním rohu obrázku, lze obrázek editovat, označovat si hvězdy určené k porovnání. Blíže je to vidět na následujícím obrázku.



*Obr. 3-9. Označení hvězd pro porovnání*

Kde: var – je označená proměnná (variabilní) hvězda  
comp – srovnávací (komparativní) hvězda  
chk1, chk2, ... – případné kontrolní hvězdy

Naměřená data, použitá v praktické části této diplomové práce patří hvězdám ze souhvězdí Vozky (lat. Auriga) (Obr. 3-10), jedná se o otevřenou hvězdokupu (*open cluster*, *OC*) označovanou NGC 2281, viz též obrázek (Obr. 3-8). Nejjasnější hvězdou je Capella.



Obr. 3-10. Snímek hvězdné oblohy se souhvězdím Vozky  
([www.jthommes.com/Astro/NGC2281.htm](http://www.jthommes.com/Astro/NGC2281.htm))

NGC je zkratkou z anglického *New General Catalogue*, jedná se o nejznámější katalog objektů hlubokého vesmíru v amatérské astronomii, jehož autorem je John Louis Emil Dreyer. Označení NGC xxx nenesou pouze hvězdokupy, ale i objekty jako galaxie,

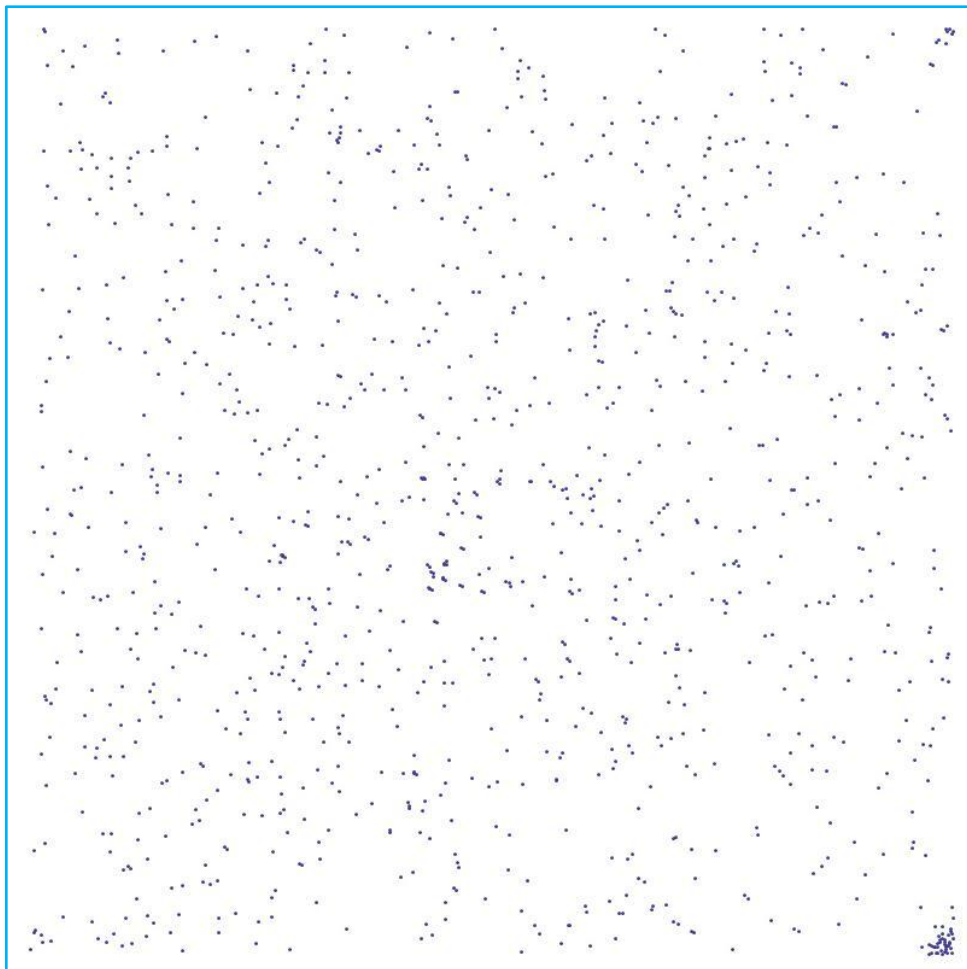
kulové hvězdokupy, mlhoviny, hvězdy, dvojhvězdy, tříhvězdí, skupiny hvězd, supernovy, zbytky supernovy a další, ovšem jsou zde zaneseny i neexistující objekty, které byly do katalogu přidány pravděpodobně omylem.

## **II. PRAKTICKÁ ČÁST**

## 4 NÁVRH PROGRAMOVÉHO ŘEŠENÍ

Cílem práce je navrhnout programové řešení pro porovnání proměnných hvězd s využitím evolučních výpočetních technik jako náhradu a usnadnění práce namísto ručního výpočtu. Programové řešení je dále aplikováno na dodané datové soubory a výsledky vyhodnoceny.

Dodané datové soubory obsahují záznamy hvězd souhvězdí Vozky (lat. Auriga). Jedná se o otevřenou hvězdokupu (*open cluster, OC*) označovanou NGC 2281. Ze souřadnic jsme schopni pomocí nějakého softwaru vykreslit podobu hvězdné mapy. Na obrázku (Obr. 4-1) je mapa vykreslená v programu Wolfram Mathematica 8.0.4, kterou lze porovnat s obrázky (Obr. 3-8) a (Obr. 3-10).



Obr. 4-1. Hvězdná mapa vykreslená v programu Mathematica

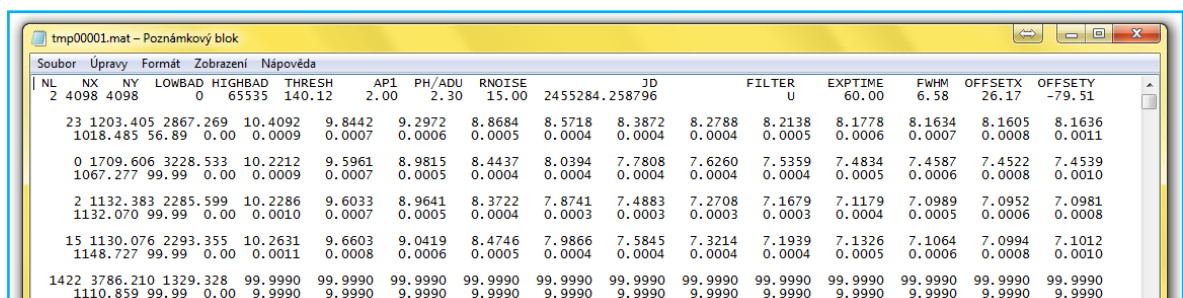
K návrhu programového řešení byl využit software *Wolfram Mathematica 8.0.4 for Students*, který je nabízen studentům Univerzity Tomáše Bati ve Zlíně pro domácí použití a to zcela zdarma na jeden rok.

Software Mathematica využívá následujícího základního rozdělení barev:

- **Funkce, příkazy, typy, hodnoty, znaménka, závorky, proměnné (již použité)**
- **Proměnné (nové, nevyužité)**
- **Ukazatel cyklu**
- **"Text"**
- **(\* Poznámky \*)**
- **Slot (#), vstupní proměnné**

#### 4.1 Převedení \*.mat na \*.txt

Jakmile jsou provedena měření jasů hvězd na určitém úseku oblohy, k čemuž se využívá CCD kamer, jsou data zpracována programem *Munimatch*, který je součástí softwarového balíku *C-Munipack* a vyexportovaná do souboru s příponou *mat*, jak lze vidět na obrázku (Obr. 4-2). Poněvadž je k návrhu programového řešení použito softwaru Mathematica, je soubor převeden to textového formátu *txt* a zároveň je odebrána hlavička souboru (první dva řádky), která pro další práci není důležitá a navíc pak soubor obsahuje pouze numerické hodnoty, čímž je následující práce ulehčena.



Soubor	Úpravy	Formát	Zobrazení	Nápověda											
1	NL	NX	NY	LOWBAD	HIGHBAD	THRESH	AP1	PH/ADU	RNOISE	JD	FILTER	EXPTIME	FWHM	OFFSETX	OFFSETY
2	4098	4098	0	65535	140.12	2.00	2.30	15.00	2455284.258796	U	60.00	6.58	26.17	-79.51	
23	1203.405	2867.269	10.4092	9.8442	9.2972	8.8684	8.5718	8.3872	8.2788	8.2138	8.1778	8.1634	8.1605	8.1636	
	1018.485	56.89	0.00	0.0009	0.0007	0.0006	0.0005	0.0004	0.0004	0.0005	0.0006	0.0007	0.0008	0.0011	
0	1709.606	3228.533	10.2212	9.5961	8.9815	8.4437	8.0394	7.7808	7.6260	7.5359	7.4834	7.4587	7.4522	7.4539	
	1067.277	99.99	0.00	0.0009	0.0007	0.0005	0.0004	0.0004	0.0004	0.0005	0.0006	0.0006	0.0008	0.0010	
2	1132.383	2285.599	10.2286	9.6033	8.9641	8.3722	7.8741	7.4883	7.2708	7.1679	7.1179	7.0989	7.0952	7.0981	
	1132.070	99.99	0.00	0.0010	0.0007	0.0005	0.0004	0.0003	0.0003	0.0003	0.0003	0.0005	0.0006	0.0008	
15	1130.076	2293.355	10.2631	9.6603	9.0419	8.4746	7.9866	7.5845	7.3214	7.1939	7.1326	7.1064	7.0994	7.1012	
	1148.727	99.99	0.00	0.0011	0.0008	0.0006	0.0005	0.0004	0.0004	0.0004	0.0005	0.0006	0.0008	0.0010	
1422	3786.210	1329.328	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	99.9990	
	1110.859	99.99	0.00	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	9.9990	

Obr. 4-2. Vstupní soubor tmp00001.mat

Každý *MAT* soubor (Obr. 4-2) obsahuje:

Tab. 4-1. Hlavička souboru *MAT*

NL	Verze formátu (vždy 2)
NX	Šířka snímku v px
NY	Výška snímku v px
LOWBAD	Nejnižší platná hodnota obrazového bodu v ADU
HIGHBAD	Nejvyšší platná hodnota obrazového bodu v ADU
THRESH	---
AP1	Poloměr clonky 1 v px
PH/ADU	Převodní poměr AD převodníku
RNOISE	---
JD	Datum a čas středu expozice
FILTER	Použitý barevný filtr
EXPTIME	Expoziční doba v sekundách
FWHM	Střední hodnota FWHM hvězd
OFFSETX	Relativní posun snímku v ose X v px
OFSETY	Relativní posun snímku v ose Y v px

Tab. 4-2. Záznamy o hvězdách

Pozice	1. řádek	2. řádek
1.	Identifikační číslo hvězdy	Jas oblohy
2.	Střed hvězdy v ose X v px	Střední chyba jasu oblohy
3.	Střed hvězdy v ose Y v px	Nepoužito (vždy 0)
4.	Instrumentální velikost hvězdy v magnitudách (clonka 1)	Chyba hvězdné velikosti (clonka 1)
...	...	...
15.	Instrumentální velikost hvězdy v magnitudách (clonka 12)	Chyba hvězdné velikosti (clonka 12)

Základem, pro práci se soubory v programu Mathematica, je nastavení cesty k souborům. Následující kód automaticky zjistí umístění souboru \*.mat se zdrojovým kódem a nastaví tuto cestu jako výchozí.

```
SetDirectory[NotebookDirectory[]]
```

```
(* Cesta k souborům *)
```

Pro návrh programového řešení je k dispozici 267 souborů, z čehož každý soubor obsahuje záznamy 2277 hvězd. Pro převedení souborů \*.mat na \*.txt slouží následující část kódu:

```
soubory = FileNames["NGC 2281\\*.mat"]; (* Jména souborů *.mat *)
pocet = Dimensions[soubory][[1]]; (* Počet souborů *.mat *)
Table[{
  jmenoVstup = FileNames["NGC 2281\\*.mat"][[i]],
                                     (* Jméno i-tého souboru .mat *)
  vstup = ReadList[jmenoVstup, String, NullRecords->True],
          (* Načtení i-tého souboru jako "text" i s prázdnými řádky *)
  dimenze = Dimensions[vstup][[1]], (* Počet řádků souboru *)
  vystup = Take[vstup, -(dimenze-2)],
          (* Odebrání prvních dvou řádků ze souboru (obsahují jen hlavičky
          sloupců v souboru) *)
  jmenoVystup = "NGC 2281\\"<>FileBaseName[jmenoVstup]<>".txt",
               (* Upravení jména souboru na .txt *)
  Export[jmenoVystup, vystup], (* Export i-tého souboru .txt *)
  Close[jmenoVystup]          (* Uzavření i-tého souboru *)
}, {i, 1, pocet}]; (* Upravení souborů *.mat na *.txt *)
```

## 4.2 Zprůměrování jasů

Abychom si mohli návrh pravidelně kontrolovat, vracet se pouze o pár kroků zpět, provádět opakovaně jen některé výpočty, budou se „i“ postupné kroky ukládat do samostatných souborů. Toto nám ulehčí práci i v případě, kdy se, například při posledním postupu, zmýlíme ve výpočtu a nebudeme tak muset provádět celý zdlouhavý postup znovu, ale vrátíme se jen k poslednímu postupu „souboru“.

Pro tyto účely si zvolíme následující soubory:

```
soubor = "Prumery.txt"; (* Jméno nového souboru s průměry hvězd *)
soubor2 = "Serazeno.txt";
          (* Jméno nového souboru se seříděnými průměry hvězd *)
soubor3 = "Porovnani_manual.txt";
          (* Jméno nového souboru s porovnanými průměry hvězd - man. výběr *)
soubor4 = "Porovnani_SOMA.txt";
          (* Jméno nového souboru s porovnanými průměry hvězd - SOMA *)
soubor5 = "Porovnani_DE.txt";
          (* Jméno nového souboru s porovnanými průměry hvězd - DE *)
```

Dále si musíme zjistit počet souborů (pocet) a počet řádků v souboru (dimStr), resp. počet hvězd v každém souboru (dimStr / 3). Jak lze vidět na obrázku (Obr. 4-2), záznam každé hvězdy se skládá ze dvou řádků dat a jednoho řádku pro oddělení záznamů.

```
soubory = FileNames["NGC 2281\*.txt"]; (* Jména souborů *.txt *)
pocet = Dimensions[soubory][[1]]; (* Počet souborů *.txt *)
jmenoVstup = soubory[[1]]; (* Jméno 1. souboru .txt *)
vstupStr = ReadList[jmenoVstup, String, NullRecords->True];
(* Načtení 1. souboru jako "text" i s prázdnými řádky *)
dimStr = Dimensions[vstupStr][[1]]; (* Počet řádků souboru *)
```

Nyní nastává proces zprůměrování jasů hvězd. Ze všech souborů se načte první řádek záznamu první hvězdy a zprůměrují se první jasy ze všech souborů, pak druhé jasy, čtvrté jasy, až dvanácté jasy ze všech souborů. Tímto získáme 12 průměrů jasů jedné hvězdy. Z těch 12i průměrů vybereme minimální a maximální jas, jejich rozdílem určíme  $\Delta$  a společně se souřadnicemi hvězdy uložíme do nového souboru (Prumery.txt). Tento postup opakujeme pro všechny hvězdy. Na začátek souboru je uložena hlavička.

```
WriteString[soubor, "P.c.\tSouradnice...\t\tPrumery jasu (12)...\n
\tX\tY\t\t1\t2\t3\t4\t5\t6\t7\t8\t9\t10\t11\t12\tDelta\n\n"];
(* 2 řádkový zápis hlavičky *)
pomRad = Table[1, {pocet}]; (* Pomocné pole 267 hodnot:
pro každý soubor buňka s číslem řádku *)

(* i ... označení pro hvězdu *)
(* j ... označení pro soubor *)
(* k ... pomocná proměnná pro cykly *)
c = 0; (* Pořadové číslo hvězdy *)
Table[{
  ++c;
  xx = yy = 0; (* Souřadnice X a Y *)
  vypocet = Table[0, {12}],
  (* Vynulování pomocného pole pro součet jasů *)
  prumer = Table[0, {12}],
  (* Vynulování pomocného pole pro počet jasů *)
  Table[{
    jmenoVstup = FileNames["NGC 2281\*.txt"][[j]],
    (* Jméno j-tého souboru .txt *)
    vstupStr = ReadList[jmenoVstup, String, NullRecords -> True],
    (* Načtení j-tého souboru jako "text" i s prázdnými řádky *)
    pom = ReadList[jmenoVstup, Number],
    (* Načtení j-tého souboru jako "čísla" bez prázdných řádků *)
    vstupNum = Partition[pom, 30],
    (* Rozdělení vstupu po 30i záznamech (30 záznamů = 1 hvězda) *)
```

```

If[vstupStr[[i]] != "", { (* Test i-tého řádku j-tého souboru,
                          zda není prázdný (na řádku jsou čísla) *)
                          (* Jsou-li čísla, počítá se průměr *)
                          (* Je-li prázdný, soubor se přeskakuje a jde se na další *)
  radek = pomRad[[j]], (* Pomocná proměnná s číslem aktuálního
                        řádku "hvězdy" v j-tém souboru pro lepší viditelnost v kódu *)
  If[xx == 0, xx = vstupNum[[radek, 2]],
    (* Test na Xovou souřadnici aktuální hvězdy.
      Není-li ještě zapsaná, pak se zapíše ze souboru *)
  If[yy == 0, yy = vstupNum[[radek, 3]],
    (* Test na Yovou souřadnici aktuální hvězdy.
      Není-li ještě zapsaná, pak se zapíše ze souboru *)

  Table[
    If[vstupNum[[radek, k + 3]] != 99.9990, {
      (* Test k-tého jasu na daném řádku, zda není roven 99.9990 *)
      (* Není-li rovný 99.9990, jde o jas a provádí se výpočet *)
      (* Je-li rovný 99.9990, nejde o jas, ale "chybu" a výpočet se
        přeskakuje *)
      (* k + 3 je tu toho důvodu, že jasy začínají až 4 číslem v řadě *)
      vypocet[[k]] = vypocet[[k]] + vstupNum[[radek, k + 3]],
      (* Součet všech k-tých jasů dané hvězdy přes všechny soubory *)
      ++prumer[[k]]
      (* V Poli "prumer" se inkrementuje k-té číslo
        (ve výsledku dá počet sečtených jasů) *)
    }
    , {k, 12}];, (* Průchod všemi 12i jasy aktuální hvězdy *)
    ++pomRad[[j]]
  (* Po průchodu celé hvězdy v j-tém souboru se posune ukazatel na
    další hvězdu v tomto souboru *)

  }
}, {j, pocet}], (* Průchod všemi soubory *)
Table[vypocet[[k]] = vypocet[[k]] / prumer[[k]], {k, 12}],
(* Samotný průměr jasů: Součet všech k-tých jasů hvězdy / jejich
  počet *)

delta = Max[vypocet] - Min[vypocet],
(* Delta = rozdíl max a min jasu *)

Table[{
  If[k == 1, WriteString[soubor, ToString[c], "\t"],
  If[k == 2, WriteString[soubor, ToString[xx], "\t"],
  If[k == 3, WriteString[soubor, ToString[yy], "\t\t"],
  If[k > 3 && k < 16, WriteString[soubor, vypocet[[k - 3]],
"\t"],
  If[k == 16, WriteString[soubor, delta, "\n"]
}, {k, 16}] (* Zápis do souboru *)
}, {i, 2, dimStr, 3}]; (* Začátek 2. řádek. Posun o 3 řádky *)
(* Procházení každého řádku s jasy pro všechny (dimStr) hvězdy *)
Close[soubor]; (* Uzavření souboru s průměry hvězd *)

```

Výsledný soubor *Prumery.txt* vypadá následovně:

P. c.	Souradnice... X	Y	Prumery jasu (12)... 1	2	3	4	5	6	7	8	9	10	11	12	Delta
1	1203.41	2867.27	10.3993	9.82016	9.27689	8.87152	8.60926	8.45329	8.3632	8.31367	8.29061	8.28186	8.28032	8.28271	2.11894
2	1709.61	3228.53	10.2796	9.66646	9.05672	8.51825	8.09719	7.83674	7.69846	7.62676	7.59306	7.58018	7.57704	7.57877	2.70256
3	1132.38	2285.6	10.2627	9.64401	9.0215	8.45998	7.98855	7.62509	7.40677	7.29985	7.25549	7.23877	7.23486	7.23669	3.02788
4	1130.08	2293.36	10.4168	9.8163	9.20496	8.63414	8.12202	7.70589	7.45062	7.31667	7.25046	7.22548	7.21908	7.22034	3.19772
5	3786.21	1329.33	10.3352	9.72002	9.09397	8.53024	8.0563	7.71303	7.52973	7.44132	7.39922	7.37934	7.36516	7.35282	2.98242
6	3791.18	1325.86	10.2385	9.62193	9.00341	8.45116	7.99388	7.67057	7.48562	7.38491	7.33316	7.31098	7.30507	7.30717	2.93347
7	1774.6	2428.76	10.2675	9.64797	9.02339	8.46222	7.98705	7.60293	7.34956	7.21789	7.16399	7.14639	7.14206	7.14407	3.12539
8	1826.59	2191.48	10.273	9.65653	9.04175	8.49939	8.06592	7.78979	7.65061	7.57849	7.54342	7.52964	7.52659	7.52879	2.7464
9	3126.64	2325.56	10.3196	9.70049	9.07357	8.51101	8.01539	7.56651	7.14712	6.74996	6.39322	6.13486	5.98781	5.89968	4.41994
10	1713.45	3227.53	10.2375	9.62674	9.01858	8.48035	8.05126	7.79238	7.66633	7.60322	7.57376	7.5626	7.5604	7.56272	2.67709
11	1643.57	2292.44	10.2571	9.64585	9.02858	8.46076	7.99626	7.63952	7.39837	7.32258	7.19801	7.02522	7.00238	6.89584	2.71138

Obr. 4-3. Soubor *Prumery.txt*

Identifikační číslo z původního souboru je zde nahrazeno pořadovým číslem (P.c.), které je dáno pořadím každé hvězdy. Každá hvězda má ve všech původních souborech stejně pořadí.

### 4.3 Seřazení průměrů dle $\Delta$

Následující krok je pouhopouhé seřazení souboru *Prumery.txt*, a to dle posledního sloupečku označeného *Delta* ( $\Delta$ ). Seřazení probíhá vzestupně, od nejmenšího. Delta jas proměnné hvězdy je rozdíl mezi maximálním a minimálním naměřeným jasem dané hvězdy, tudíž nejmenší  $\Delta$  jas označuje nejkonstantněji svítící hvězdu.

První část kódu načte soubor *Prumery.txt*, odstraní hlavičku a uloží soubor do dočasného souboru *temp.txt*.

```
vstup = ReadList[soubor, String, NullRecords->True];
(* Načtení souboru "Prumery.txt" jako "text" i s prázdnými řádky *)
dimenze = Dimensions[vstup][[1]];
vystup = Take[vstup, -(dimenze-3)];
(* Odstraní první 3 řádků *)
souborT = "temp.txt";
(* Jméno dočasného souboru s průměry hvězd *)
Export[souborT, vystup];
```

Druhá část pak provádí samotné seřazení obsahu souboru *temp.txt* dle sloupce  $\Delta$  jasu a výsledek uloží do souboru *Serazeno.txt*.

```

head = Take[vstup, 2]; (* Načtení hlavičky souboru "Prumery.txt" *)
vstup = ReadList[souborT, Number];
(* Načtení souboru "temp.txt" jako "čísla" *)
DeleteFile[souborT]; (* Smazání dočasného souboru *)
pom = Partition[vstup, 16];
(* Rozdělení vstupu po 16i záznamech (16 záznamů = 1 hvězda) *)
dimenze = Dimensions[pom][[1]];
rozdily = Table[{pom[[i,1]], pom[[i,16]]}, {i, dimenze};
(* Matice: 1. sloupec = číslo hvězdy, 2. sloupec = delta jas *)
setrideno = Sort[rozdily, #1[[2]] < #2[[2]]&];
(* Seříděná předchozí matice dle 2. sloupce *)
Table[WriteString[soubor2, "\t"<>head[[i]]<>"\n"], {i, 2}];
(* Vypsání hlavičky do souboru "Serazeno.txt" *)
WriteString[soubor2, "\n"];
Table[{
  radek = setrideno[[i,1]],
  (* Číslo nejkonstantnější hvězdy "řádku" *)
  vysledek = pom[[radek]],
  (* Záznam dané hvězdy *)
  Table[{
    If[j == 1, WriteString[soubor2, ToString[i], "\t"],
    If[j ≤ 2, WriteString[soubor2, vysledek[[j]], "\t"],
    If[j == 3, WriteString[soubor2, vysledek[[j]], "\t\t"],
    If[j > 3 && j < 16, WriteString[soubor2, vysledek[[j]], "\t"],
    If[j == 16, WriteString[soubor2, vysledek[[j]], "\n"]
  }, {j, 16}]
  (* Zápis záznamu do souboru *)
}, {i, dimenze}];
(* Zápis do souboru pro všechny hvězdy *)
Close[soubor2]; (* Uzavření souboru se seřazenými průměry hvězd *)

```

Na následujícím obrázku (Obr. 4-4) je výřez výsledného souboru *Serazeno.txt*, chybějící sloupce jsou nahrazeny (...). První sloupec je aktuální pořadí v souboru, druhý sloupec jsou pořadová čísla hvězd v původním souboru.

	P. c.	Souradnice...		Prumery jasu (12)...			...	11	12	Delta
		X	Y	1	2	3				
1	44	1037.29	1846.24	10.8191	10.2682	9.75002	...	8.78131	8.78476	2.03782
2	52	1913.25	2415.22	11.1303	10.5795	10.0615	...	9.08711	9.08972	2.04324
3	46	1264.91	2514.35	11.0118	10.4578	9.93546	...	8.96815	8.97062	2.04365
4	55	1495.19	2209.74	11.2096	10.6579	10.1382	...	9.1647	9.16775	2.04489
5	59	912.205	1327.71	11.3562	10.8162	10.3097	...	9.29952	9.30095	2.05667
6	60	1441.36	1156.35	11.3652	10.8219	10.3117	...	9.30614	9.30947	2.05904
7	38	1374.36	1505.13	10.7821	10.2315	9.71221	...	8.72078	8.72227	2.06136
8	53	2012.3	2442.28	11.184	10.6258	10.0972	...	9.11739	9.11991	2.06662
9	43	1994.01	2117.18	10.8326	10.2767	9.75049	...	8.76345	8.76601	2.06918
10	37	1802.42	2574.15	10.7289	10.1715	9.64353	...	8.65887	8.66152	2.07003
11	58	2053.21	2738.39	11.3151	10.7539	10.2216	...	9.24315	9.24439	2.07194

Obr. 4-4. Výřez ze souboru *Serazeno.txt*

#### 4.4 Nominální hvězda

Jakmile jsou hvězdy seřazeny dle  $\Delta$  jasů, přichází na řadu určení tzv. *Nominální hvězdy*. Nominální hvězdou se rozumí fiktivní hvězda složená z  $n$  vybraných *referenčních* hvězd.

První část kódu načte soubor *Serazeno.txt*, odstraní hlavičku a uloží soubor do dočasného souboru *temp.txt*.

```
vstup = ReadList[soubor2, String, NullRecords->True];
dimenze = Dimensions[vstup][[1]];
vystup = Take[vstup, -(dimenze-3)];
souborT = "temp.txt";
Export[souborT, vystup];
head = Take[vstup, 2]; (* Načtení hlavičky souboru "Serazeno.txt" *)
```

Druhá část provádí výpočet nominální hvězdy, který je následující: z prostředku souboru *temp.txt* se vyberou průměry jasů  $n$  hvězd, vypočítají se celkové průměry všech 12i jasů a ze získaných 12i průměrů se opět určí maximální a minimální průměrný jas a jejich rozdílem je získán  $\Delta$  jas nominální hvězdy.

```
n = 5; (* Počet hvězd, ze kterých se bude počítat nominální *)
pom = ReadList[souborT, Number];
DeleteFile[souborT]; (* Smazání dočasného souboru *)
vstup = Partition[pom, 17];
(* Rozdělení vstupu po 17i záznamech (17 záznamů = 1 hvězda) *)
dimenze = Dimensions[vstup][[1]];
Nom = Ceiling[(dimenze-n+1) / 2]; (* První pro výpočet nominální *)
VyberN = Table[{{vstup[[i,1]], vstup[[i,2]]}, {i, Nom, Nom+n-1}};
(* Matice: 1. Sloupec = pořadové číslo, 2.s = číslo hvězdy *)
JasNH = Table[Mean[vstup[[#[[1]] &/@VyberN, i]]], {i, 5, 16}];
(* 12 průměrů jasů z n hvězd Nominální Hvězdy *)
AppendTo[JasNH, Max[JasNH] - Min[JasNH]]; (* Doplnění o deltu *)
```

Manuální výběr referenčních hvězd byl proveden 10x, ať už z prostředních hvězd (předešlá část kódu), tak i nahodile, výsledky jsou uvedeny v tabulce (Tab. 4-3).

Po vypočtení nominální hvězdy se do souboru *Porovnaní\_manual.txt* uloží referenční hvězdy, hvězda nominální a následně i ostatní hvězdy se souřadnicemi a porovnanými  $\Delta$  jasy s  $\Delta$  jasem nominální hvězdy.

```

WriteString[soubor3, "Vyber:\n-----\n\n"];
Table[WriteString[soubor3, head[[i]]<>"\n"], {i, 2}];
(* Vypsání hlavičky do souboru "Porovnaní.txt" *)
WriteString[soubor3, "\n"];
Table[{
  radek = VyberN[[i, 1]],
  (* Číslo i-té hvězdy, ze které se počítá nominální *)

  zaznam = vstup[[radek]],
  (* Záznam dané hvězdy *)
  Table[{
    If[j ≤ 3, WriteString[soubor3, zaznam[[j]], "\t"],
    If[j == 4, WriteString[soubor3, zaznam[[j]], "\t\t"],
    If[j > 4 && j < 17, WriteString[soubor3, zaznam[[j]], "\t"],
    If[j == 17, WriteString[soubor3, zaznam[[j]], "\n"]
  ], {j, 17}]]
}, {i, n}];
(* Zápís do souboru pro všechny hvězdy *)
Table[{
  If[j == 1, WriteString[soubor3, "\nNominalni\t\t\t\t\t"],
  If[j < 13, WriteString[soubor3, JasNH[[j]], "\t"],
  If[j == 13, WriteString[soubor3, JasNH[[j]],
"\n\n\nPorovnaní:\n-----\n\n\tP.c.\tX\tY\t\tDelta\n\n"]
}, {j, 13}];
(* Zápís záznamu do souboru *)
Table[{
  If[!MemberQ#[[1]] &/@VyberN, i],
  zaznam = vstup[[i]];
  (* Záznam dané hvězdy *)
  Table[{
    If[j ≤ 3, WriteString[soubor3, zaznam[[j]], "\t"],
    If[j == 4, WriteString[soubor3, zaznam[[j]], "\t\t"],
    If[j == 5, WriteString[soubor3,
      JasNH[[Dimensions[JasNH][[1]]]] -
      zaznam[[Dimensions[zaznam][[1]]]], "\n"]
  ], {j, 5}]]
}, {i, dimenze}];
(* Zápís do souboru pro všechny hvězdy *)
Close[soubor3];

```

Výsledný soubor pak vypadá následovně:

Vyber	P. c.	Souradnice...	Prumery jasu (12)...	4	5	6	7	8	9	10	11	12	Delta
28	57	2935.5 3917.07	11.285 10.7048 10.1367 9.69672 9.39985 9.21974 9.11124 9.04264 9.00974 8.99789 8.99029 8.96773 2.31725										
29	47	2078.03 1968.36	11.0616 10.5064 9.98108 9.58171 9.31812 9.15411 9.03438 8.88677 8.70344 8.58455 8.53479 8.51985 2.5417										
30	15	1840.45 2385.64	10.2735 9.66018 9.05309 8.5303 8.14761 7.93164 7.81932 7.76131 7.73417 7.724 7.72218 7.72429 2.55137										
31	17	1840.48 2389.44	10.3436 9.72777 9.12166 8.59763 8.21383 7.98043 7.84609 7.7719 7.73651 7.7235 7.72078 7.72267 2.62281										
32	12	1784.47 2358.72	10.3599 9.77819 9.23493 8.82969 8.56775 8.41175 8.31838 8.25399 8.18123 8.0228 7.80404 7.70894 2.65098										
Nominalni			10.6647 10.0755 9.50549 9.04721 8.72943 8.53953 8.42588 8.34332 8.27302 8.21055 8.15442 8.1287 2.53602										
	P. c.	X Y	Delta										
1	44	1037.29 1846.24	0.498204										
2	52	1913.25 2415.22	0.492784										
3	46	1264.91 2514.35	0.492374										
4	55	1495.19 2209.74	0.491134										
5	59	912.205 1327.71	0.479354										
6	60	1441.36 1156.35	0.476984										
7	38	1374.36 1505.13	0.474664										
8	53	2012.3 2442.28	0.469404										
9	43	1994.01 2117.18	0.466844										
10	37	1802.42 2574.15	0.465994										
11	58	2053.21 2738.39	0.464084										

Obr. 4-5. Soubor Porovnaní.txt

## 4.5 Aplikace evoluce

V předešlém postupu se nominální hvězda tvořila z  $n$  prostředních hvězd, tzv. referenčních. Aplikací evolučního algoritmu tento krok výběru vypustíme a ideální kombinaci  $n$  referenčních hvězd vyhledá samotný evoluční algoritmus, např. SOMA.

Jinak řečeno, výsledkem běhu evolučního algoritmu jsou vybrané referenční hvězdy, které tvoří nejkvalitnější kombinaci pro porovnání proměnlivosti hvězdy.

Následující část kódu, stejně jako v předchozím postupu, načte soubor *Serazeno.txt*, odstraní hlavičku a uloží soubor do dočasného souboru *temp.txt*.

```
n = 5;          (* Počet hvězd, ze kterých se bude počítat nominální *)
vstup = ReadList[soubor3, String, NullRecords->True];
dimenze = Dimensions[vstup][[1]];
vystup = Take[vstup, -(dimenze-3)];
souborT = "temp.txt";
Export[souborT, vystup];
head = Take[vstup, 2]; (* Načtení hlavičky souboru "Serazeno.txt" *)
pom = ReadList[souborT, Number];
DeleteFile[souborT];          (* Smazání dočasného souboru *)
vstup = Partition[pom, 17];
(* Rozdělení vstupu po 17i záznamech (17 záznamů = 1 hvězda) *)
dimenze = Dimensions[vstup][[1]];
```

Před aplikací samotného evolučního algoritmu je nutné nadefinovat vzorového jedince, tzv. *specimena* a účelovou funkci (*CostFunction*), dle které se budou jedinci ohodnocovat.

```
Specimen = Table[{{Integer, {1, dimenze}}}, {n}]
```

*Specimen* je  $n$  rozměrný jedinec, kde každý prvek *specimena* je celé číslo z intervalu  $\{1, \text{dimenze}\}$ , kde „*dimenze*“ je dána počtem hvězd, ze kterých se hledá nominální hvězda.

```
CostFunction = Compile[{{x, _Integer, 1}},
  x2 = Round[x];
  JasNH = Table[Mean[vstup[[Table[x2][[i]], {i, 1,
    Dimensions[x2][[1]]}], j]], {j, 5, 16}]; (* Jas Nominální Hvězdy *)
  If[Plus @@ (Equal[#1[[1]], #1[[2]] & /@ (Permutations[
    x2, {2}])) /. {True -> 1, False -> 0}) == 0,
    Max[JasNH] - Min[JasNH],
    (Max[JasNH] - Min[JasNH])*100*(Plus @@ x2)
  ]]
```

Jelikož optimalizační metody pracují v oblasti reálných čísel a parametry jedince jsou čísla celá, jsou parametry jedince zaokrouhlovány již před ohodnocením účelovou funkcí.

Pro eliminaci chybových jedinců populace je v účelové funkci přidána podmínka:

```
Plus @@ ((Equal[#1[[1]], #1[[2]]] & /@ (Permutations[x, {2}]))
/. {True -> 1, False -> 0}) == 0
```

kteřá testuje, zda jsou parametry jedince různá celá čísla. Je-li podmínka splněna, je jedinec tvořen různými číslicemi a účelová funkce vrací hodnotu účelové funkce, kterou je ve výsledku delta jas nominální hvězdy. Obsahuje-li jedinec minimálně dva prvky stejné, je hodnota účelové funkce tohoto jedince penalizována vynásobením čísla 100 a součtem prvků jedince, čímž se dostane mimo uvažované adepty do další generace.

Pro nalezení referenčních hvězd, kromě manuálního výběru, byly vyzkoušeny algoritmy *SOMA* a *Diferenciální evoluce*.

#### 4.5.1 SOMA

Následující část kódu, 5 podprogramů, je volně stažitelná z internetových stránek (<http://www.fai.utb.cz/people/zelinka/soma/>).

Vygenerování prvopočáteční populace.

```
DoPopulation[popsiz_ , Specimen_] := Module[{pop, cv},
  pop = Table[
    MapThread[Random[#1[[#2, 1]], #1[[#2, 2]]] &,
      {Specimen, Random[Integer, {1, #1}] & /@
        Flatten[{#1} & /@ (Dimensions[#1][[1]] & /@ Specimen)]}],
    {i, popsiz};
  cv = CostFunction /@ pop;
  Return[MapThread[{#1, #2} &, {cv, pop}]]
]
```

Samotný SOMA algoritmus.

```

AOSOMA[Pop_] :=
MapIndexed[#1[[2, #1[[1, 1, 1]]]] &,
  ({Position[#1[[1]], Min[#1[[1]]]], #1[[2]]} &/@
  (({#1[[1]] &/@ #1}, #1} &/@
  (Map[{CostFunction[#1], #1} &,
  (Table[
    Flatten[MapIndexed[CheckInterval,
      #1[[2]] + (Pop[[
        Position[Pop, Min[#1[[1]] &/@ Pop]][[1, 1]]
        , 2]]
      - #1[[2]]) t (Table[
        If[RandomReal[] < PRT, 1, 0], {i, Dim}]])
    , {t, 0, PathLength, Step}] &/@ Pop
  )
  , {2}))
  ))
)
]

CheckInterval[IndVal_, Pos_] := Module[{LogInt},
  LogInt = IntervalMemberQ[Interval[#1[[2]]], IndVal] &/@
  Specimen[[Pos[[1]]]];
  Return[If[MemberQ[LogInt, True], IndVal,
  Random[Specimen[[Pos[[1]], #1, 1]],
  Specimen[[Pos[[1]], #1, 2]]] &/@
  {Random[Integer, {1, Dimensions[Specimen[[Pos[[1]]]][[1]]]}]
  ]}
]

```

Tvar výstupu algoritmu SOMA.

```

ExpForm[nmbr_] := PaddedForm[nmbr, {6, 5},
  ExponentFunction -> (#1 &),
  NumberFormat -> (#1 <> "<E>" <> #3 &),
  NumberSigns -> {"-", "+"}];

BestInd[pop_] := Module[{best, ind, str},
  best = Position[##, Min[##]][[1]][[1]] & @@
  Transpose[pop][[1]];
  ind = pop[[best]];
  str = "Best individual is on position " <> ToString[best] <>
  " with cost value " <> ToString[ExpForm[ind[[1]]]] <>
  " and parameters " <> ToString[Table[ToString[ExpForm[ind[[2,
  i]]], {i, Length[ind[[2]]}]]];
  Print[str];
  Return[Flatten[{best, ind}, 1]]
]

```

Před samotným spuštěním evolučního algoritmu je nutné nastavit parametry potřebné k běhu algoritmu. Parametry *PopSize* a *Migrations* jsou nastavovány, dle uvážení uživatele, nastavení parametrů *Step*, *PathLength* a *PRT* je doporučeno a vychází z dlouhého testování. Parametr *MinDiv* je v tomto případě nepodstatný, proto je nastaven na zápornou hodnotu.

```
Dim = Dimensions[Specimen][[1]];          (* Rozměr jedince *)
PopSize = 30;                             (* Velikost populace *)
Migrations = 50;                          (* Počet migračních kol *)
Step = .22;                               (* Krok jedince *)
PathLength = 3.;                          (* Délka cesty jedince *)
PRT = .1;                                  (* Perturbační vektor *)
MinDiv = -1.; (* Max. rozdíl mezi nejlepším a nejhorším jedincem *)
```

Dle kapitoly 1.1 „Základní dogma EVT“ je druhým krokem vygenerování prvopočáteční populace. Ta je vytvořena dle specimena a obsahuje tolik jedinců, jak je nastaven parametr *PopSize*.

```
Population = DoPopulation[PopSize, Specimen]
```

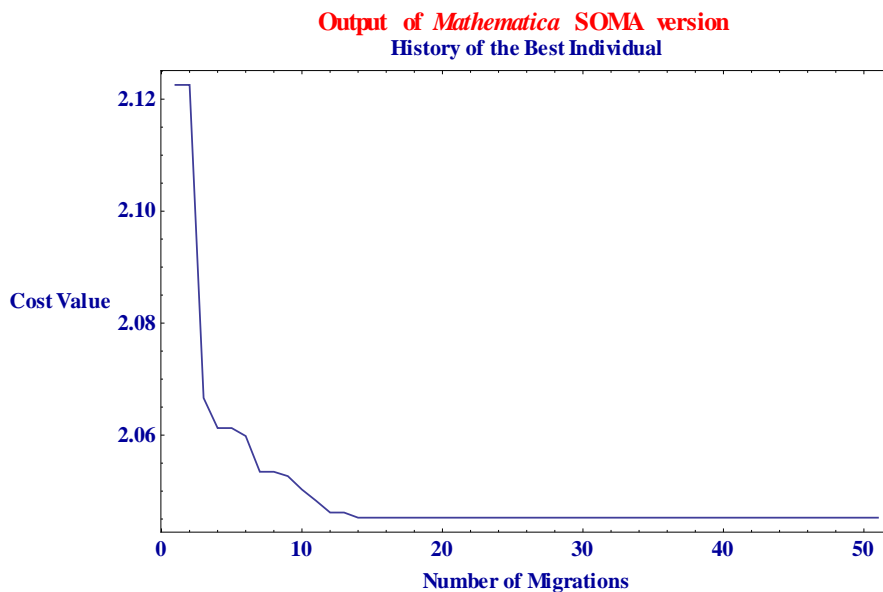
Nyní nastává běh samotného evolučního algoritmu. Na podprogram *AOSOMA* je aplikovaná populace *Population* při počtu migrací *Migrations*.

```
FinalPopulation = NestList[AOSOMA, Population, Migrations];
BestInd[Last[FinalPopulation]]
```

Výstup algoritmu je pak následující:

```
Best individual is on position 2 with cost value +2.04524<E>0 and
parameters {+5.20000<E>1, +4.58433<E>1, +3.70000<E>1,
+6.00000<E>1, +5.86690<E>1}
{2, 2.04524, {52, 46, 37, 60, 59}}
```

Běh nejlepšího jedince v tomto případě lze pak vidět na následujícím obrázku.



Obr. 4-6. Běh nejlepšího jedince algoritmu SOMA

Algoritmus SOMA byl spuštěn 10x, vždy na nové počáteční populaci, výsledky jsou uvedeny v tabulce (Tab. 4-3).

Podobně, jako v případě manuálního výběru, se do souboru *Porovnaní\_SOMA.txt* uloží příklad vybraných referenčních hvězd, hvězda nominální a následně i ostatní hvězdy se souřadnicemi a porovnanými  $\Delta$  jasy s  $\Delta$  jasem nominální hvězdy. K tomu se využije následující část kódu, která si uchová i průměrné jasy hvězdy nominální a výsledek se uloží do souboru (*soubor4 = Porovnaní\_SOMA.txt*). Výsledný soubor pak vypadá stejně, jako na obrázku (Obr. 4-5).

```
JasNH = Table[Mean[vstup[[# &/@ VyberN, i]]], {i, 4, 15}]
(* Jas Nominální Hvězdy *)
AppendTo[JasNH, Max[JasNH] - Min[JasNH]] (* Doplnění o deltu *)
```

#### 4.5.2 Diferenciální evoluce

Následující část kódu, 6 podprogramů, je volně stažitelná z internetových stránek (<http://www.fai.utb.cz/people/zelinka/soma/>).

Samotný DE algoritmus.

```

DERandlBin[Pop_] := MapThread[If[#1[[1]] < #2[[1]], #1, #2] &,
  {Pop,
    ({CostFunction[#1], #1} &/@
      (Flatten[MapIndexed[CheckInterval[#1, #2] &, #1], 1] &/@
        (Flatten[Table[If[Cr < Random[], Pop[[i, 2, j]],
          #1[[i, j]], {i, NP}, {j, Dim}] &/@
            {F*(#1[[1, 2]] - #1[[2, 2]]) + #1[[3, 2]] &/@
              (Pop[[#1]] &/@ MapIndexed[SelectOther[#2[[1]]] &,
                Pop]], 1))))))}
]

CheckInterval[IndVal_, Pos_] := Module[{LogInt},
  LogInt = IntervalMemberQ[Interval[#1[[2]]], IndVal] &/@
  Specimen[[Pos[[1]]]];
  Return[If[MemberQ[LogInt, True], IndVal,
    Random[Specimen[[Pos[[1]], #1, 1]],
      Specimen[[Pos[[1]], #1, 2]]] &/@
    {Random[Integer, {1, Dimensions[Specimen[[Pos[[1]]]]][[1]]}]}
  ]
]

SelectOther[active_] := Module[{},
  rand = {0, 0, 0};
  While[rand[[1]] == rand[[2]] || rand[[1]] == rand[[3]] ||
    rand[[2]] == rand[[3]] || active == rand[[1]] ||
    active == rand[[2]] || active == rand[[3]],
    rand = Table[Random[Integer, {1, NP}], {i, 3}];
  Return[rand]
]

```

Podprogramy, pro tvorbu prvopočáteční populace a pro tvar výstupu algoritmu DE, jsou stejné jako u algoritmu SOMA.

Před samotným spuštěním diferenciální evoluce je nutné nastavit parametry potřebné k běhu algoritmu. Parametry *NP* a *Generations* jsou nastavovány, dle uvážení uživatele, nastavení parametrů *F* a *Cr* je doporučené a vychází z dlouhého testování.

```

Dim = Dimensions[Specimen][[1]];          (* Rozměr jedince *)
NP = 30;                                  (* Velikost populace *)
Generations = 50;                          (* Počet generací *)
F = 0.8;                                   (* Mutační konstanta *)
Cr = 0.2;                                  (* Práh křížení *)

```

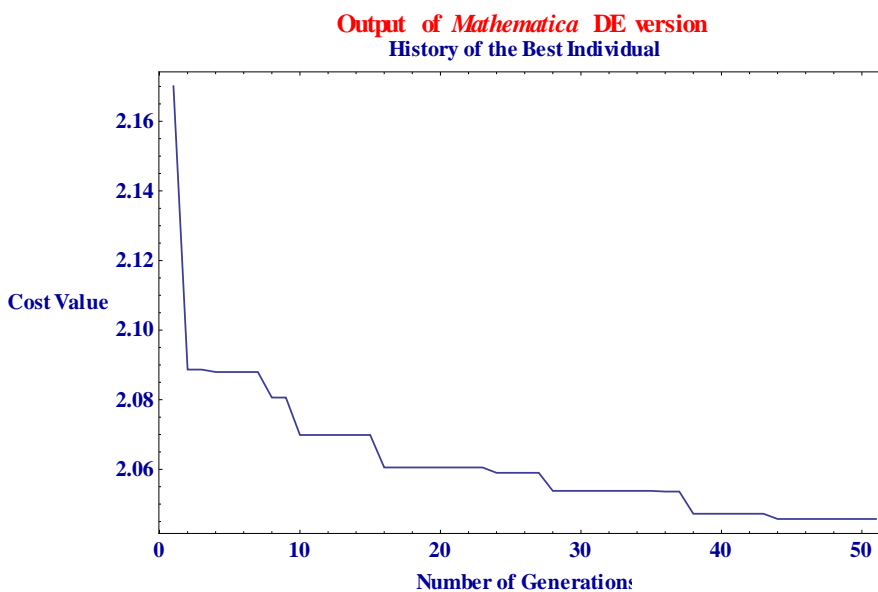
Prvopočáteční populace je vytvořena dle specimena a obsahuje tolik jedinců, jak je nastaven parametr  $NP$ .

```
Population = DoPopulation[NP, Specimen]
```

Nyní nastává běh samotného evolučního algoritmu. Na podprogram *DERandlBin* je aplikovaná populace *Population* při počtu migrací *Generations*.

```
FinalPopulation = NestList[DERandlBin, Population, Generations];
BestInd[Last[FinalPopulation]]
```

Výstup algoritmu je pak stejného charakteru, jako u algoritmu SOMA. Běh nejlepšího jedince v tomto případě lze pak vidět na následujícím obrázku.



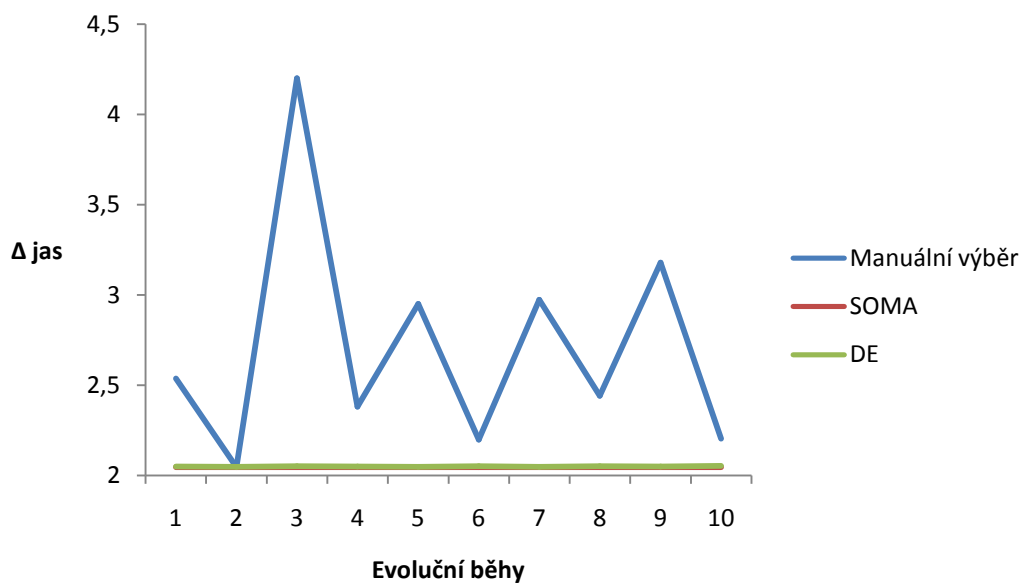
Obr. 4-7. Běh nejlepšího jedince diferenciální evoluce

Diferenciální evoluce byla spuštěna 10x, vždy na nové počáteční populaci, výsledky jsou uvedeny v tabulce (Tab. 4-3).

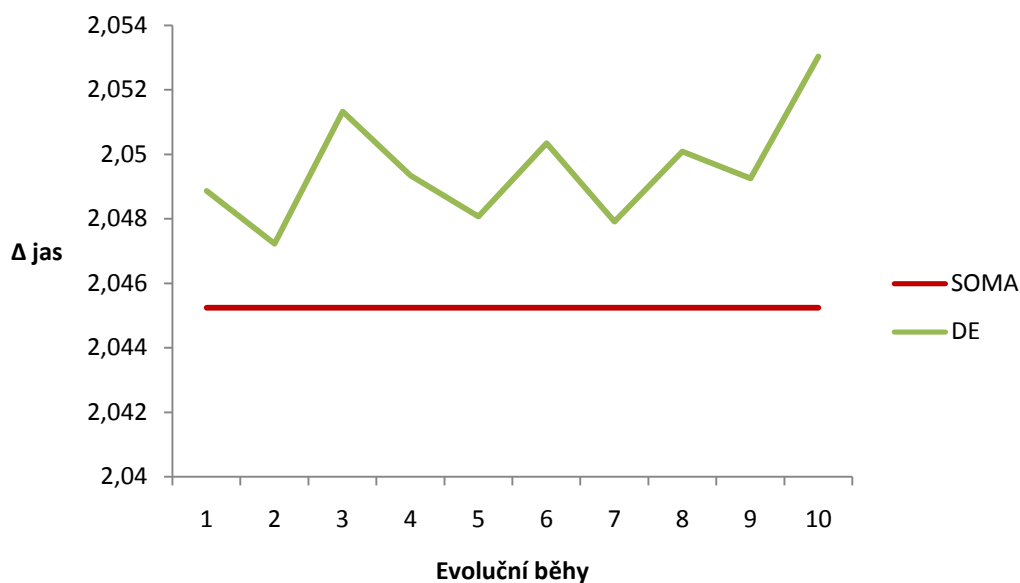
Podobně, jako v případě SOMY, se do souboru (*soubor5 = Porovnani\_DE.txt*) uloží příklad vybraných referenčních hvězd, hvězda nominální a následně i ostatní hvězdy se souřadnicemi a porovnanými  $\Delta$  jasy s  $\Delta$  jasem nominální hvězdy. Výsledný soubor pak vypadá stejně, jako na obrázku (Obr. 4-5).

Tab. 4-3. Výsledky manuálního výběru a běhu evolucí ze seřazeného souboru (Serazeno.txt)

Manuální výběr		SOMA			Diferenciální evoluce		
$\Delta$ jas	Hvězdy	Jedinec	$\Delta$ jas	Hvězdy	Jedinec	$\Delta$ jas	Hvězdy
2.53602	28, 29, 30, 31, 32	1	2.04524	1, 4, 5, 3, 2	9	2.04887	3, 1, 7, 4, 5
2.04524	1, 2, 3, 4, 5	1	2.04524	5, 4, 3, 1, 2	9	2.04723	8, 2, 3, 4, 1
4.19898	56, 57, 58, 59, 60	1	2.04524	4, 1, 2, 3, 5	11	2.05132	2, 3, 1, 15, 4
2.38035	3, 54, 24, 7, 10	1	2.04524	3, 2, 4, 1, 5	4	2.04934	1, 7, 4, 3, 6
2.95032	60, 44, 39, 8, 24	1	2.04524	4, 1, 5, 2, 3	28	2.04807	6, 1, 5, 2, 3
2.87	56, 38, 21, 52, 5	1	2.04524	1, 5, 3, 4, 2	24	2.05034	5, 4, 1, 9, 2
2.97285	1, 21, 41, 51, 60	1	2.04524	1, 4, 2, 5, 3	14	2.04791	10, 4, 1, 3, 2
2.44127	5, 12, 46, 38, 9	1	2.04524	4, 1, 3, 5, 2	14	2.05009	5, 9, 2, 3, 1
3.17842	7, 42, 31, 56, 59	1	2.04524	2, 4, 5, 3, 1	5	2.04925	7, 4, 6, 1, 2
2.27	23, 41, 12, 4, 44	1	2.04524	3, 1, 5, 2, 4	28	2.05303	6, 2, 5, 7, 4



Obr. 4-8. Porovnání  $\Delta$  jasů manuálního výběru a EVT ze seřazeného souboru (Serazeno.txt)

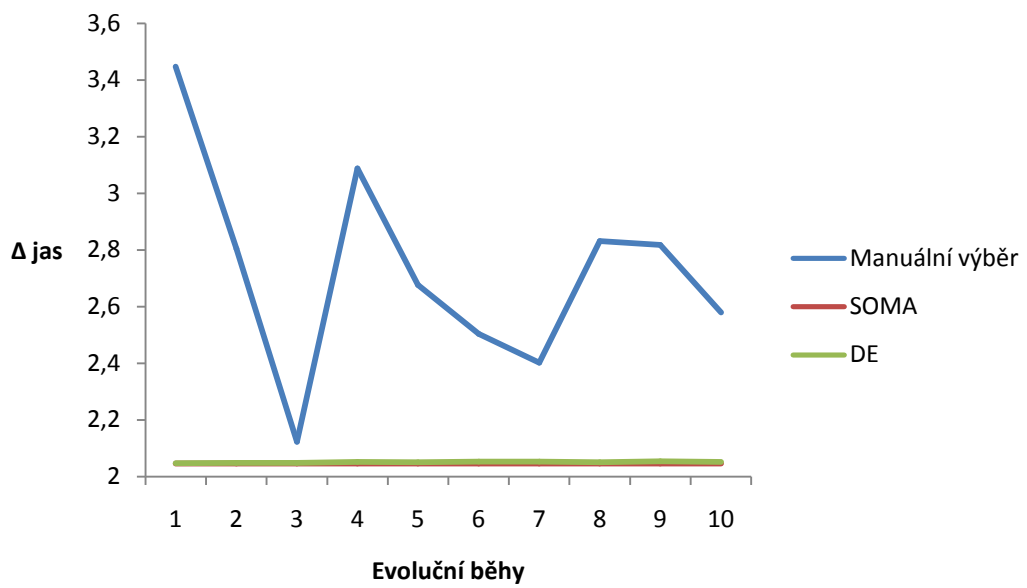


Obr. 4-9. Porovnání vypočítaných  $\Delta$  jasů SOMY a DE ze seřazeného souboru (Serazeno.txt)

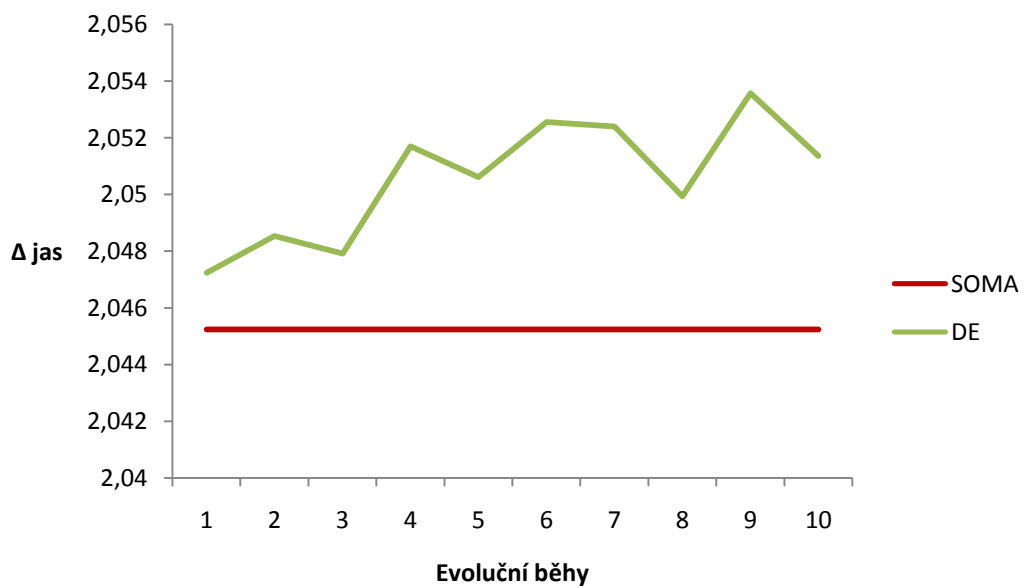
Pro získání výsledků byl použit soubor (*soubor2 = Serazeno.txt*) se seřazenými hvězdami dle  $\Delta$  jasů. Proto byly předchozí postupy lehce upraveny, pro využití souboru (*soubor = Prumery.txt*) s neseřazenými hvězdami a postup opakován. Získané výsledky jsou uvedeny v tabulce (Tab. 4-4).

Tab. 4-4. Výsledky manuálního výběru a běhu evolucí z neseřazeného souboru (Prumery.txt)

Manuální výběr		SOMA			Diferenciální evoluce		
$\Delta$ jas	Hvězdy	Jedinec	$\Delta$ jas	Hvězdy	Jedinec	$\Delta$ jas	Hvězdy
3.44653	28, 29, 30, 31, 32	10	2.04524	59, 52, 46, 44, 55	5	2.04723	53, 52, 44, 55, 46
2.80445	1, 2, 3, 4, 5	1	2.04524	55, 59, 44, 46, 52	6	2.04853	46, 52, 59, 44, 38
2.12243	56, 57, 58, 59, 60	1	2.04524	52, 59, 55, 46, 44	30	2.04791	37, 46, 55, 52, 44
3.0878	3, 54, 24, 7, 10	4	2.04524	59, 55, 44, 52, 46	27	2.05169	46, 52, 37, 59, 55
2.67655	60, 44, 39, 8, 24	1	2.04524	44, 55, 46, 52, 59	13	2.05061	46, 37, 55, 59, 44
2.50357	56, 38, 21, 52, 5	2	2.04524	46, 44, 59, 52, 55	13	2.05255	46, 55, 52, 39, 59
2.40127	1, 21, 41, 51, 60	1	2.04524	46, 52, 55, 44, 59	28	2.0524	55, 46, 39, 38, 44
2.83085	5, 12, 46, 38, 9	2	2.04524	52, 46, 59, 44, 55	18	2.04993	44, 55, 59, 53, 46
2.81805	7, 42, 31, 56, 59	1	2.04524	55, 44, 52, 59, 46	16	2.05357	37, 58, 55, 52, 44
2.57962	23, 41, 12, 4, 44	1	2.04524	46, 59, 44, 52, 55	30	2.05136	43, 55, 46, 38, 44



Obr. 4-10. Porovnání  $\Delta$  jasů manuálního výběru a EVT z neseřazeného souboru (Prumery.txt)



Obr. 4-11. Porovnání vypočítaných  $\Delta$  jasů SOMY a DE z neseřazeného souboru (Prumery.txt)

## ZÁVĚR

Hlavní myšlenou pro vznik této práce bylo využití evolučních výpočetních technik v astronomii, přesněji pro identifikaci proměnných hvězd.

V teoretické části je práce zaměřená nejen na rozdělení a stručný popis funkčnosti základních optimalizačních a evolučních algoritmů, ale i na popis základních pojmů optimalizace. Protože se jedná o využití EVT při identifikaci proměnných hvězd, je zde také popsáno rozdělení proměnných hvězd, jejich stručný popis doplněný o zajímavé obrázky, převážně z oblasti supernov, které jsou zajímavé tím, že se jedná o „konec života“ hvězdy. Pojem *supernova* lze volně přeložit, jako *exploze*. Jakmile dojde k chemické změně jádra hvězdy (supernova typu II), či ke zvýšení hmotnosti hvězdy nad kritickou hranici 1.3 – 1.4 násobku Slunce (supernova typu Ia), případně jejich kombinací (supernova typu Ib a Ic), dojde k explozi hvězdy a vzniká vesmírná mlhovina, tzv. *supernova*. Tu lze na obloze pozorovat pomocí hvězdářských dalekohledů a teleskopů až několik měsíců, dle typu supernovy. V teoretické části také nechybí stručný popis přístrojů a softwarových produktů sloužících k pozorování a měření hvězd.

Ve druhé, praktické části se pak práce zabývá popisem navrhovaného programového řešení pro identifikaci proměnných hvězd. Samotné aplikaci evolučních technik předbíhalo několik úprav dodaných datových souborů, výpočtů a komunikací se specialisty v oblasti astronomie. K výběru referenčních hvězd, které tvoří hvězdu nominální, se přistupovalo ze dvou pohledů, a to přístupem manuálního výběru referenčních hvězd a využitím EVT k nalezení referenčních hvězd. K tomuto účelu byly testovány dvě evoluční techniky, SOMA a diferenciální evoluce. Manuální výběr i obě EVT byly testovány na souboru s hvězdami, které byly seřazeny dle  $\Delta$  jasů, a poté i na souboru, kde tyto hvězdy nebyly seřazeny a bylo tudíž těžší získat optimální kombinaci referenčních hvězd. V obou případech, ačkoli byla diferenciální evoluce rychlejší, SOMA vykazovala kvalitnější řešení problému. Získané výsledky jsou uvedeny v tabulkách a pro přehlednost i v grafech.

## CONCLUSION

The main thought for creation of this work was application of evolutionary computing in astronomy, especially in identification of variable stars.

In the theoretical part is this work focused on the distribution and description of functionality of the evolution algorithm and on the description of optimizations basic terms. This work is about the usage of EVT for identification of variable stars, so there is described distribution of variable stars, their descriptions which are complemented with interesting pictures. These pictures are largely of the supernovas. The supernovas are interesting because supernova is the “end of life” of the stars. We can translate the term *supernova* as *explosion*. When the core of star gets chemical changes (supernova II) or the weight of the star is over critical limit, which is 1.3– 1.4 of the weight of the Sun, (supernova Ia) or in case on their combination (supernova Ib and Ic) the star will explode and it will create space nebula called a *supernova*. This nebula can be seen with telescopes for a few months, depending of the type of supernova. The theoretical part of this work includes description of the devices and software products which can be used for observing and measuring of stars.

In the second, practical, part is this work focused on description of the proposed software solution for the identification of variable stars. Some adjusting of data files, calculations and communications with specialists of astronomy was done before the application of evolutionary techniques. To choose reference stars, which form nominal star, was approached from two views – approach with manual choose of reference stars and EVT use for finding reference stars. It was testing for that reason two evolution techniques - SOMA and Diferencial evolution. Manual choosing and both EVT techniques was testing on stars file, which was sorted by  $\Delta$  of brightness and than was testing also on file, where this stars was not sorted and in this case was more difficult to gain optimal combination of reference stars. In both cases, although Diferencial evolution was faster, SOMA had more quality of problem sorting. Acquired results are shown in tables and for transparency also in graphs.

**SEZNAM POUŽITÉ LITERATURY**

- [1] BRÁT, Luboš, Ladislav ŠMELCER a Jaroslav TRNKA. *Proměnné hvězdy a možnosti jejich pozorování a výzkumu*. Valašské Meziříčí, 2011. Dostupné z: [http://www.astrovm.cz/userfiles/file/projekty/kosoap/vzdelavaci\\_metodicky\\_material\\_KOSOAP-PH-maly.pdf](http://www.astrovm.cz/userfiles/file/projekty/kosoap/vzdelavaci_metodicky_material_KOSOAP-PH-maly.pdf)
- [2] HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008. ISBN 978-80-247-2695-3 (brož.).
- [3] KVASNIČKA V., POSPÍCHAL J., TIŇO P., *Evoluční algoritmy*, STU Bratislava, 2000, 215 s. ISBN 80-227-1377-5
- [4] MOTL, David. *C-Munipack 1.1: Uživatelský manuál*. 16. dubna 2008. Dostupné z: <ftp://ftp.heanet.ie/mirrors/sourceforge/c/project/c-/c-munipack/C-Munipack%201.1%20Documentation/1.9/cmpack-1.9-doc-cz.pdf>
- [5] RECHENBERG I., *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis)*, 1971. Fromman-Holzboog, 1973.
- [6] TVRDÍK, Josef. *Evoluční algoritmy*. Ostrava, 2004. Dostupné z: [http://prf.osu.cz/doktorske\\_studium/dokumenty/Evolutionary\\_Algorithms.pdf](http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf).  
Učební text. Ostravská univerzita
- [7] ZELINKA, Ivan, OPLATKOVÁ Zuzana, ŠEDA Miloš, OŠMERA Pavel a VČELARĚ František. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.
- [8] ZELINKA, Ivan. *Evolutionary algorithms and chaotic systems*. Berlin : Springer, 2010. 521 s. ISBN 978-3-642-10707-8.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ACO	Ant Colony Optimization (Optimalizace mravenčí kolonií)
BT	Bílý trpaslík, pojmenování menší složky dvojhvězdy
$C^k$	ohodnocení cesty $k$ -tého mravence u ACO
CCD	Charge-Coupled Device, kamery pracující na principu fotoefektu
CR	práh křížení, řídicí parametr $\epsilon$ (0, 1) u DE
comp	označení pro srovnávací (komparativní) hvězdu v programu C-Munipack
D	počet argumentů účelové funkce u algoritmu SOMA a DE
DE	Diferenciální evoluce
ES	Evoluční strategie (Evolutionary Strategy)
EVT	Evoluční výpočetní techniky (Evolutionary Computing)
$f(x)$	označení pro účelovou funkci
$f_{\text{cost}}(x)$	označení pro účelovou funkci, nazývanou jako cenová funkce
$f_{\text{max}}(x)$	označení účelové funkce, jejíž optimalizace vede k nalezení maxima
$f_{\text{min}}(x)$	označení účelové funkce, jejíž optimalizace vede k nalezení minima
F	mutační konstanta, řídicí parametr $\epsilon$ (0, 2) u DE
FV	vhodnost (Fitness Value)
G	označení generace u DE
GA	Genetické algoritmy (Genetic Algorithms)
Generations	počet generací, ukončovací parametr DE
HC	Horolezecký algoritmus (Hill Climbing)
HeII	jedenkrát ionizované atomy helia
HeIII	zcela ionizované atomy helia
Hi	horní hranice specimena
HST	Hubble Space Telescope (Hubbleův vesmírný dalekohled)

chk1, chk2,...	označení pro případné kontrolní hvězdy v programu C-Munipack
$k$	označení mravence u algoritmu ACO
Lo	dolní hranice specimena
M	počet jedinců v populaci
$M$	prostor řešení funkce, příp. označení migrace u algoritmu SOMA
$m$	značí rozdíl mezi pozicí vedoucího jedince $x_{L,j}^M$ a startovní pozici j-tého prvku i-tého jedince $x_{i,j,start}^M$ v migračním kole $M$ , příp. počet mravenců u algoritmu ACO
MAT	přípona souborů programu C-Munipack, jsou-li výsledkem skládání
Migrace	počet migračních kol, ukončovací parametr algoritmu SOMA
MinDiv	Minimal Diversity, ukončovací parametr algoritmu SOMA
N	počet parametrů jedince
$N(x,\sigma)$	množina sousedních řešení $x \in M$
NGC	New General Catalogue, katalog objektů vesmíru v amatérské astronomii
NP	velikost populace, minimálně 4 jedinci, řídicí parametr DE
$n_T$	počet opakování <i>Metropolisova algoritmu</i> pro danou teplotu u SA
PathLength	délka cesty, řídicí parametr algoritmu SOMA
PopSize	velikost populace, řídicí parametr algoritmu SOMA
PRT	perturbace, parametr $\in (0, 1)$ , dle které se generuje perturbační vektor, řídicí parametr algoritmu SOMA
PRTVektor	perturbační vektor, řídí pohyb jedince u algoritmu SOMA
PRTVektor <sub>j</sub>	j-tý prvek perturbačního vektoru pro j-tého potomka jedince
$Q$	množství feromonu uložené mravencem u ACO
$r$	náhodně vygenerované číslo z intervalu $(0, S)$ při výběru rodičů u GA
$r$	viz. $x_{i,j}^{M+1}$
$r_0$	viz. $x_{i,j,start}^M$

$r_1, r_2, r_3, r_4$	označení vybraných rodičů DE pro generování nové populace potomků
$rnd$	náhodně generované číslo z intervalu (0, 1) pro generování perturbačního vektoru u algoritmu SOMA, příp. pro sestavení zkušební vektoru u DE
$S$	suma hodnot účelových funkcí v populaci při výběru rodičů u GA
SA	Simulované žihání (Simulated Annealing)
SOMA	SamoOrganizující se Migrační Algoritmus
SRT	přípona souboru programu C-Munipack, je-li soubor výstupem fotometrie
STEP	krok, řídicí parametr algoritmu SOMA
$T_0$	počáteční teplota u SA
$T_f$	konečná teplota, tzv. krystalizační u SA
$t_{max}$	zvolený počet iterací u SA
$v_s$	šumový vektor u DE
$v_{s,j}$	j-tý prvek šumového vektoru
$v_z$	zkušební vektor u DE
$v_{z,j}$	j-tý prvek zkušební vektoru
var	označení pro proměnnou (variabilní) hvězdu v programu C-Munipack
$x_0$	počáteční řešení $x_0 \in M$
$x^i$	i-ty jedinec u ES
$x_i$	prvopočáteční náhodně vygenerovaný rodič u ES
$x_{i,j}^{M+1}$	značí j-tý prvek i-tého jedince v migračním kole $M+1$ u algoritmu SOMA
$x_{i,j, start}^M$	značí startovní pozici j-tého prvku i-tého jedince v migračním kole $M$
$x_{r_1,j}^G, x_{r_2,j}^G, x_{r_3,j}^G, x_{r_4,j}^G$	j-tý prvek vektoru rodiče $r_1, r_2, r_3$ , příp. $r_4$ v generaci $G$
$y_{rek}$	rekombinant u ES
$\alpha$	funkce redukce teploty u SA, příp. parametr kontroly vlivu na $\tau_{ij}$ u ACO
$\beta$	parametr kontroly vlivu na $\eta_{ij}$ u ACO

---

$\eta_{ij}$	požadavek na cestu mezi body $i$ a $j$ u ACO
$\lambda$	označení pro populaci potomků u ES
$\mu$	označení pro populaci rodičů u ES
$\rho$	rekombinační parametr u ES, příp. poměr vypařování feromonu u ACO
$\sigma$	směrodatná odchylka pro Gaussovo normální rozdělení u ES
$\tau_{ij}$	feromonová stopa mezi body $i$ a $j$ u ACO
$\Delta\tau_{ij}^k$	množství naneseného feromonu $k$ -tým mravencem mezi body $i$ a $j$ u ACO

## SEZNAM OBRÁZKŮ

Obr. 1-1. Rozdělení optimalizačních metod .....	16
Obr. 2-1. Příklad běhu slepého algoritmu s červeně naznačeným průběhem nejlepších dosažených řešení.....	18
Obr. 2-2. Příklad běhu horolezeckého algoritmu pro unimodální funkci.....	21
Obr. 2-3. Příklad algoritmu simulovaného žíhání.....	23
Obr. 2-4. Příklad jednobodového křížení.....	24
Obr. 2-5. Příklad dvoubodového křížení .....	25
Obr. 2-6. Příklad vícebodového křížení.....	25
Obr. 2-7. Příklad jednobodové a vícebodové mutace .....	25
Obr. 3-1. Akreční disk při přetoku hmoty dvojhvězdy.....	37
Obr. 3-2. Supernova.....	39
Obr. 3-3. Zbytky supernovy (NGC 6960).....	39
Obr. 3-4. Supernova pravděpodobně typu Ia.....	40
Obr. 3-5. Pozůstatek supernovy, jejíž světlo roku 1680 zasáhlo i Zemi.....	40
Obr. 3-6. Supernova SN2006aj.....	41
Obr. 3-7. Rozdělení proměnných hvězd a soustav hvězd.....	41
Obr. 3-8. Otevřená hvězdokupa NGC 2281 v programu C-Munipack.....	43
Obr. 3-9. Označení hvězd pro porovnání.....	43
Obr. 3-10. Snímek hvězdné oblohy se souhvězdím Vozky.....	44
Obr. 4-1. Hvězdná mapa vykreslená v programu Mathematica .....	47
Obr. 4-2. Vstupní soubor tmp00001.mat .....	48
Obr. 4-3. Soubor Prumery.txt .....	53
Obr. 4-4. Výřez ze souboru Serazeno.txt.....	54
Obr. 4-5. Soubor Porovnani.txt.....	56
Obr. 4-6. Běh nejlepšího jedince algoritmu SOMA .....	61
Obr. 4-7. Běh nejlepšího jedince diferenciální evoluce.....	63
Obr. 4-8. Porovnání $\Delta$ jasů manuálního výběru a EVT ze seřazeného souboru (Serazeno.txt) .....	64
Obr. 4-9. Porovnání vypočítaných $\Delta$ jasů SOMY a DE ze seřazeného souboru (Serazeno.txt) .....	65

---

Obr. 4-10. Porovnání $\Delta$ jasů manuálního výběru a EVT z neseřazeného souboru (Prumery.txt) .....	66
Obr. 4-11. Porovnání vypočítaných $\Delta$ jasů SOMY a DE z neseřazeného souboru (Prumery.txt) .....	66

**SEZNAM TABULEK**

Tab. 4-1. Hlavička souboru MAT.....	49
Tab. 4-2. Záznamy o hvězdách.....	49
Tab. 4-3. Výsledky manuálního výběru a běhu evolucí ze seřazeného souboru (Serazeno.txt) .....	64
Tab. 4-4. Výsledky manuálního výběru a běhu evolucí z neseřazeného souboru (Prumery.txt) .....	65