Internetové rozhraní pro evoluční algoritmy a prostředí webMathematica

Internet interface for evolutionary algorithms and environment webMathematica

Bc. Tomáš Mikl

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav řízení procesů
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš MIKL**
Studijní program:	**N 2807 Chemické a procesní inženýrství**
Studijní obor:	**Automatizace a řídicí technika**
Téma práce:	**Internetové rozhraní pro evoluční algoritmy a prostředí webMathematica**

Zásady pro vypracování:

Cílem je vytvořit internetové rozhraní pro evoluční algoritmy, které by využívalo jádro prostředí webMathematica. Součástí bude i ukázka použití evolučních algoritmů.

1. Seznamte se s prostředím Mathematica a webMathematica
2. Popište propojení prostředí webMathematica s internetem
3. Zdůrazněte bezpečnostní záležitosti použití webMathematica
4. Vytvořte design webového rozhraní pro evoluční algoritmy
5. Vyberte ukázkové evoluční algoritmy, které v prostředí webMathematica naprogramujete a použijete s vhodnými optimalizačními funkcemi
6. Závěrem zhodnoťte výhody použití evolučních algoritmů v prostředí webMathematica

Rozsah práce:

Rozsah přiloh:

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

Seznam odborné literatury:

1. Kvasnička V., Pospíchal J., Tiňo P., Evolučné algoritmy, STU Bralislava, 2000, ISBN 80-227-1377-5
2. Zelinka I., Umělá inteligence v problémech globální optimalizace, BEN, 2002, 190 p. ISBN 80-7300-069-5
3. Wickham -- Jones T., webMathematica: A User guide, version 2.2., březen 2005
4. Flanagan D., JavaScript Kompletní průvodce, 2. aktualizované vydání, Computer Press, 2002, Praha, ISBN: 8072266268

Vedoucí diplomové práce: **Ing. Zuzana Oplatková**
Ústav aplikované informatiky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **24. května 2007**

Ve Zlíně dne 13. února 2007

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Petr Dostál, CSc.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá tvorbou evolučních algoritmů v prostředí webMathematica. Teoretická část práce nejprve rozebírá problematiku dynamiky webových stránek na straně klienta a serveru a technologie s tím související.

V další části jsou popsány použité funkce a názorné ukázky rozdílu mezi funkcemi v prostředí Mathematica a webMathematica, které byly využity pro evoluční algoritmy. Praktická část se věnuje popisu použitých evolučních algoritmů – Samo-Organizující Se Migrační Algoritmus (SOMA) a Diferenciální Evoluce (DE). Tyto algoritmy mají svůj teoretický popis i ve vlastních stránkách, jejichž design byl rovněž součástí zadání.

Mezi výsledky pak patří vlastní design stránek včetně ukázek výstupů jednotlivých algoritmů na testovacích funkcích.

Klíčová slova:

Mathematica, webMathematica, evoluční algoritmy, internetové rozhraní

ABSTRACT

Master thesis deals with design of evolutionary algorithms in environment webMathematica. Firstly, theoretical part of the thesis analyses problems of dynamics web sites on the side of klient and server and technologies associated with. Next part describes used functions and clearly explains a difference between functions in environment Mathematica and webMathematica, which were used for creation of evolutionary algorithms. The practical part is concerned to description of used evolutionary algorithms - Self-Organizing Migrating Algorithm (SOMA) and Differential evolution (DE). These algorithms have own theoretical description also in webpages themselves, whose design was a part of the thesis assignment too. Design of webpages belongs between results as well as preview of output of all algorithms on benchmark functions.

Keywords:

Mathematica, webMathematica, evolutionary algorithms, internet interface

Děkuji vedoucímu diplomové práce, slečně Ing. Zuzaně Oplatkové, za její odborné vedení, cenné rady a připomínky, jež mi ochotně poskytovala při řešení této práce.V neposlední řadě děkuji své rodině za podporu při studiu a za finanční zabezpečení.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

K nejmodernějším metodám globální optimalizace patří evoluční algoritmy [1]. Jejich charakteristickým rysem je, že způsob, kterým se metoda přibližuje k hledanému optimu, nalezení globálního extrému ne lokálního, je, že řešitel „nemusí“ znát klasické optimalizační metody, ale je po něm požadována pouze velmi dobrá znalost optimalizované problematiky s schopností správně nadefinovat účelovou funkci, jejíž optimalizace by měla vést k vyřešení problému. Princip evolučních algoritmů je snadno srozumitelný i nematematikům, a právě díky tomu se v průběhu posledních let velmi rychle rozšířilo jejich používání při řešení optimalizačních úloh v nejrůznějších oborech. [2]

Z matematického hlediska jsou však evoluční algoritmy pouze dalšími zástupci stochastických optimalizačních algoritmů a vyžadují řešení podobných problémů jako další stochastické optimalizační algoritmy.

Evoluční algoritmy zpracované pro softwareový program webMathematica a vytvoření internetového rozhraní pro přístup a práci s evolučními algoritmy byly zvoleny z důvodů jednoduchého a rychlého použití a prezentace evolučních algoritmů na počítačích, kde není program Mathematica a tím mít možnost zpřístupnit je široké veřejnosti.

Program webMathematica umožňuje jednoduchý způsob jak vytvářet interaktivní výpočty na webu a využít výpočetní a vizualizační schopnosti prostředí Mathematica a tím poskytnout přesné ukázky výsledků výpočtů evolučních algoritmů.

V rámci diplomové práce bude vytvořeno jednoduché internetové rozhraní pro ovládání evolučních algoritmů s tím, že uživatelé nemusí znát všechny vnitřní výpočty a funkce, ale jen základní znalosti v evolučních algoritmech SamoOrganizujícího se migračního algortimu (SOMA) a Diferenciální Evoluce (DE) [2].

Výsledkem diplomové práce bude internetové rozhraní pro evoluční algoritmy s porovnáním dvou algoritmů SOMA a DE a možnost jednoduše přidávat další algoritmy.

I. TEORETICKÁ ČÁST

1 PROSTŘEDÍ MATHEMATICA A WEBMATHEMATICA

1.1 Mathematica

Program Mathematica je software pro počítání čehokoliv vás napadne, umí počítat integrály, derivace, maticové počty, sumy a s možností zobrazení 3D grafů, disponuje obsáhlou nápovědou s kompletním seznamem všech příkazů, včetně mnoha ukázek použití.

Je to integrovaný počítačový systém, který v sobě zahrnuje možnosti programovacího jazyka, softwaru pro analytické zpracování matematických výrazů, grafického programu, numerických knihoven a v posledních verzích i textového editoru. To vše většinou jednodušeji než bývá zvykem u zmíněných jednoúčelových softwarových produktů.

Tím ovšem posunuje ono pro většinu lidí dosud poněkud obskurní programátorské řemeslo k řadám uživatelů, kteří sice umí využívat komerční software, ale tvorbu vlastních aplikací považují za nadměrně komplikovanou.

Efektivní systém pro realizaci matematiky na počítači, pracuje nejen s numerickými, ale i s algebraickými výpočty. Zahrnuje množství funkcí a algoritmů z oblasti algebry, kombinatoriky, maticového a tenzorového počtu, statistiky a regresní analýzy atd.

Je možno ho používat nejen jako „vědeckou kalkulačku“, ale umožňuje ve svém vlastním jazyce psát i samostatné programy, které jsou přenositelné mezi všemi platformami operačních systémů.

Silnou stránkou je velmi dobrá grafika, od realizace běžných grafů až po 3D zobrazování obecných parametrických ploch. Formát souborů tzv. zápisníků (notebook, přípona *.nb) je užitečný při vytváření dokumentů, které jsou nezávislé na platformě a které může sdílet více studentů, učitelů či kolegů.

Zápisníky se osvědčují při prezentaci seminářů, přednášek a názorných ukázek. Protože zobrazují tradiční matematický zápis, hodí se dobře k sestavování sylabů, testů a úloh, které se mohou buď tisknout nebo rozdávat, řešit a sbírat elektronicky.

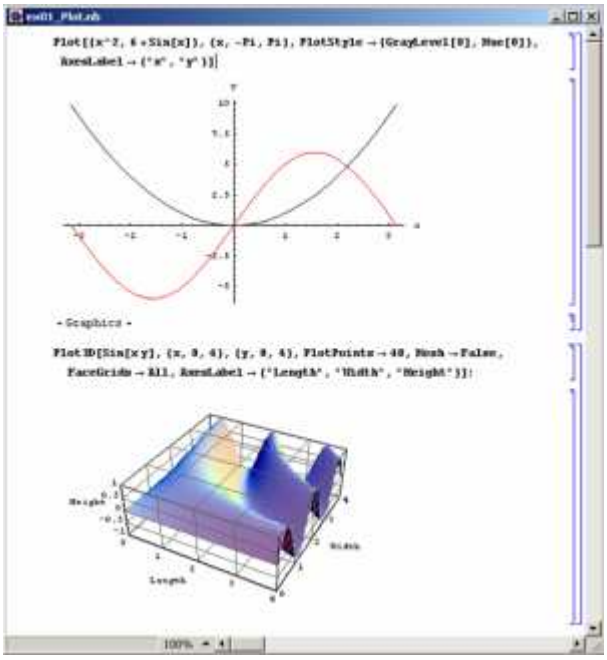
Mathematica rozšířená o sadu nástrojů navrženou speciálně pro učitele matematiky k vytváření úloh a testů ke zkouškám se výborně hodí k tvorbě podkladů pro přednášky, ke znázornění oživených ukázek ve třídě a k využití interaktivního výukového software.

1.2 webMathematica

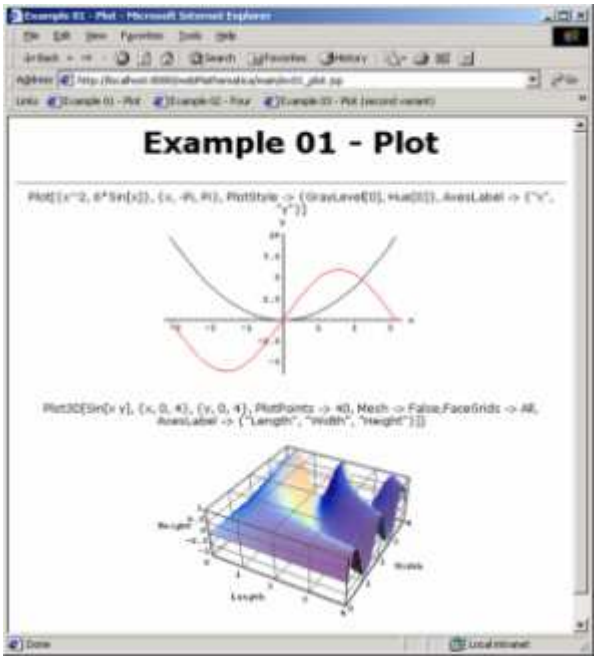
1.2.1 Co je to webMathematica

webMathematica [3] je jednoduchý způsob jak vytvářet interaktivní výpočty na webu, umožňuje vytvářet webové stránky, kde jsou využity výpočetní a vizualizační schopnosti Mathematica v celé její šíři, každému uživateli stačí standardní internetový prohlížeč pro zadávaní výpočtů i vizualizaci výsledků.

Jak je patrno z obrázků 1 a 2, vizualizace grafů je v notebooku Mathematica stejná jako na webové stránce vygenerované prostředím webMathematica.



Obr. 1 – Mathematica notebook



Obr. 2 – webMathematica stránka

1.2.2 Technologie

webMathematica je založena na dvou standardních Java technologiích – Java Servlet a Java Server Pages.

1.2.2.1 Java Servlet

Servlety jsou speciální programy v Javě, které se pouštějí přes webový server, obvykle se nazývají "servlet container", někdy "servlet engine".

Servlety jsou neoddělitelnou součástí platformy J2EE od firmy SUN. Jsou to programové komponenty běžící na straně serveru, napsané v Javě. To znamená, že vyžadují "java-enabled-server" a samozřejmě JVM (Java Virtual Machine) [4].

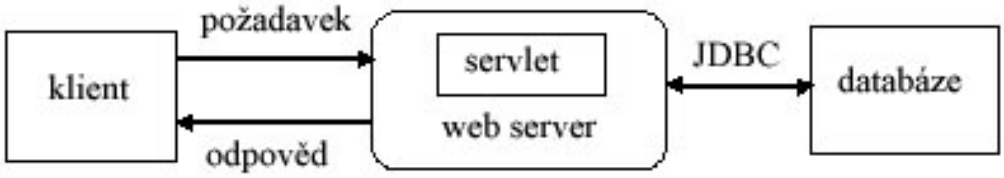
Se servery jako mezičlánkem se můžete nepřímo střetnout při technologii dynamických stránek Java Server Pages. Servlety jsou komponenty nezávislé na protokolu a platformě, napsané v Javě. Jsou vykonávané na straně serveru, přičemž musí jít o server podporující Javu.

Servlety jsou načítané a spouštěné pomocí JVM na příslušném aplikačním serveru. Podle specifikace jsou servlety nevizuální komponenty, tedy se nezobrazují pomocí žádného grafického rozhraní - GUI. Avšak to neznamená, že servlety nemohou "generovat"

uživatelské rozhraní. Ve skutečnosti se většinou používají servlety pracující nad internetovým protokolem HTTP.

Odpovědí takovýchto servletů je potom v převážné většině HTML kód. Servlety můžeme použít na různé úlohy: čtení/zapisovaní do databáze, posílaní emailů, zpracování formulářů, posílaní cookies atd. Jednoduše jsou určené na tvorbu dynamických internet/intranet řešení, podobně jako Java Server Pages.[4]

Servlety implementují princip požadavku/odpovědi, typický pro architekturu klient-server. Java Servlet API vymezuje standardní rozhraní pro obsluhu těchto požadavků a odpovědí mezi klientem a serverem. Následující obrázek obr. 3 popisuje tuto komunikaci.



Obr. 3 – Procesy mezi klientem a serverem

Klient pošle požadavek na server.

Server přepošle požadavek na konkrétní servlet.

Servlet vytvoří odpověď a předá ji serveru. (Odpověď je vytvořená dynamicky a její obsah závisí od požadavku, který poslal klient. Servlet může využít různé externí zdroje.)

Server pošle klientovi odpověď.

Servlety by jsme mohli přirovnat k CGI skriptem v tom, že pomocí servletu můžeme též vytvářet dynamický obsah.

Avšak servlety mají určité nesporné výhody v porovnaní s CGI:

Portabilita a platformová nezávislost - tuhle vlastnost mají díky jazyku Java. Java Servlet API definuje totiž standardní rozhraní mezi servletem a web serverem.

Prezentace a výkonnost - servlet je zkompilovaný a nahraný do paměti jen jednou a volaný při každém požadavku od klienta. To znamená, že servlet míň vytěžuje systémové zdroje, databázové připojení a podobně. Naproti tomu CGI je nahrané do paměti při každém požadavku.

Založeno na Javě - každé servlety jsou postavené na jazyku Java, mají k dispozici všechny výhody tohoto jazyka - objektovost, modularitu, bezpečnost a podobně.

Java Servlet API je množinou Java tříd definující rozhraní mezi klientem a servletem. Přes toto rozhraní obsluhuje servlet příchozí požadavky. Servlet API je standardní součástí Java frameworku, ale je k dispozici buď jako samostatný balíček nebo jako součást J2EE. Java Servlet API se skládá ze dvou balíčků:

javax.servlet

javax.servlet.http

Základní balíček tříd javax.servlet obsahuje abstraktní třídy a rozhraní na tvorbu všeobecných servletů pro různé protokoly, tj. že servlety mohou být použité pro libovolné protokoly (HTTP, HTTPS, FTP), záleží na konkrétní implementaci. Balík javax.servlet.http rozšiřuje funkcionalitu základního balíčku a přidává podporu protokolu HTTP.

První balík, který obsahuje mnoho různých rozhraní. Nejdůležitější je rozhraní `Servlet`. Toto rozhraní definuje metody, které musí každý servlet obsahovat. Jde hlavně o metodu service(), která obsluhuje požadavky [4].

Například konkrétní třída GenericServlet implementuje toto rozhraní. Pokud potřebujeme použít servlet pracující s protokolem HTTP, využijeme speciálnější třídu HttpServlet.

Tato třída poskytuje další metody na zpracování HTTP požadavků, například doGet() a doPost() na zpracování požadavků typu GET a POST.

Interakce klient-servlet

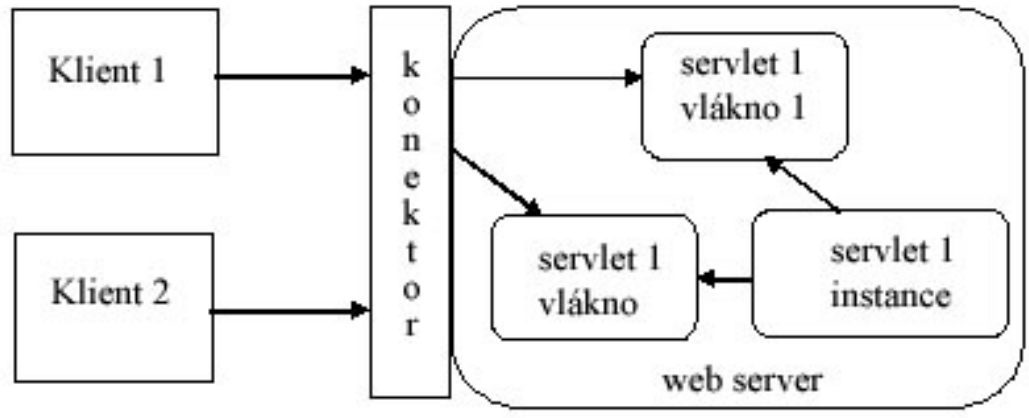
Klient, který posílá požadavek nějaké aplikaci založené na servletech, nekomunikuje přímo s nimi, ale s příslušným serverem. Server potom předá požadavek servletu pomocí Java

Servlet API. Úlohou servera je řídit spouštění a inicializovat servlet, ukončovat a likvidovat servlet z paměti.

Standardně se v paměti serveru nachází jen jeden příklad konkrétního servletu, kde zůstává i po dobu nečinnosti. Záleží na nastavení serveru, kdy bude uvolněná z paměti. Pokud server řeší více požadavků současně, server vytvoří pro každý požadavek nové vlákno.

To znamená, že servlety jsou multithreadové. Za vytvoření nového vlákna je zodpovědný server, dále je zodpovědný za načítání a znovu načítání (při změně) servletu do paměti [4].

Následující obrázek 4 zjednodušeně ilustruje základní interakci mezi klientem a servletem:



Obr. 4 – Základní interakce mezi klientem a servletem

Na začátku při prvním požadavku (nebo při změně) je servlet načítaný serverem. Všechny proměnné příklady jsou zinicializovány. Servlet zůstává aktivní přes celou dobu životnosti.

Pokud dva klienti v jeden okamžik žádají servlet o komunikaci, server vytvoří dvě nezávislá vlákna, přičemž každé vlákno má přístup k instanci servletu, jeho metodám a proměnným.

Každé vlákno obsluhuje vlastní požadavky a odezvy, které vrací příslušnému klientovi.

1.2.2.2 Java Server Pages (JSP)

Při použití JSP se podobně jako v ASP nebo v PHP přímo do HTML kódu zapisují příkazy. Ty jsou nyní zapisovány v jazyce Java. Speciální servlet se stará o to, aby byla JSP stránka vždy po své modifikaci automaticky přeložena standardního class souboru (Java byte-code) [4].

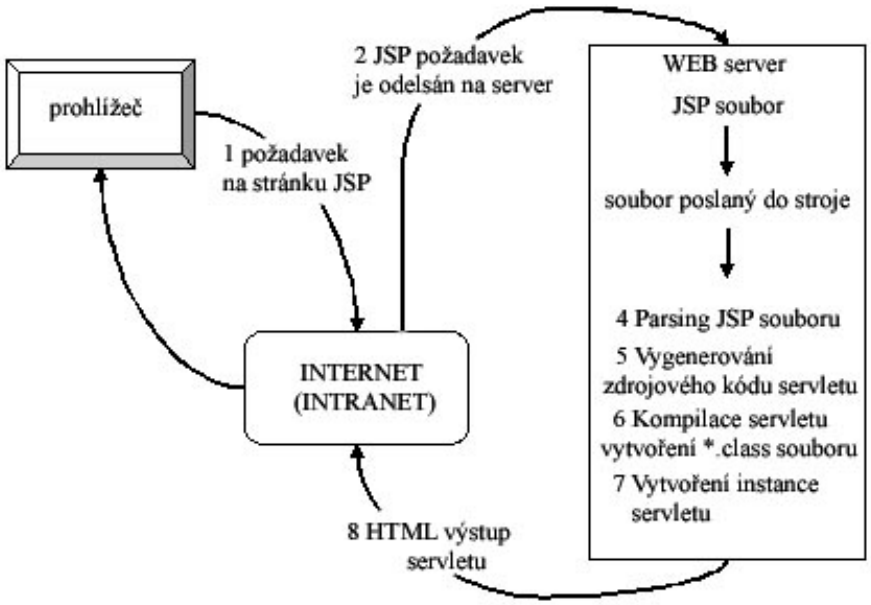
JSP byla vyvinutá společností Sun Microsystems, za účelem vývoje aplikací na straně serveru. JSP soubory jsou vlastně HTML stránky, do kterých jsou vložené speciální tagy obsahující javovský zdrojový kód. Tento kód potom vytváří dynamický obsah stránky.

To znamená, že napíšete váš HTML dokument způsobem a s nástroji, jako jste byli doteď zvyklí. Potom uzavřete kód určený pro dynamickou část do speciálních značek (tagů - většina z nich začíná **<%** a končí **%>**).

Použít JSP je vhodné hlavně, když vaši část stránky tvoří statický obsah. V opačném případě je vhodné použít na vytváření stránek technologii servletů.

1.2.3 Princip činnosti

Zdrojový kód stránky JSP je překládaný na servlet a následně kompilovaný. Výsledkem je servlet, který generuje HTML. To je posláno klientovi, jako odpověď na jeho požadavek. Detailní postup je popsaný na obr. 6.



Obr. 5 – Princip činnosti webMathematica

1. Uživatel požádá o webovou stránku, která byla vytvořená jako JSP.

2. Klient vytvoří požadavek (žádost) a směřuje ji přes síť na server.

3. Požadavek je nasměrován na příslušný Web server.

4. Web server zjistí, že požadovaný soubor je speciální, protože má koncovku "jsp". Proto přesměruje JSP soubor do JSP Servlet stroje.

5. V případě, že tento soubor byl volaný poprvé, JSP stroj ho překontroluje. V opačném případe pokračuje bodem 7.

6. V dalším kroku je vygenerovaný speciální servlet, vytvořený na základě souboru JSP. Všechny statické HTML jsou uložené v `out.println()` příkazech.

7. Zdrojový kód servletu je zkompilovaný a je vytvořený "*.class" soubor.

8. Je vytvořená instance servletu a zavolají se metody `init() a service()`.

9. Výstupem ze servletu je HTML kód, který je poslaný přes síť.

Uživateli se zobrazí výsledky.

Vzorová stránka: *datum.jsp*

Na této jednoduché stránce vidíte propojenost klasického HTML a Java kódu, uzavřeného mezi tagy <%= a %>. Příkaz `new java.util.Date()` vytvoří instanci třídy `Date` a vloží ji do stránky.

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional //EN">
 <HTML>
  <HEAD><TITLE>Dnešní datum.</TITLE></HEAD>
  <BODY>
    Dnešní datum:<BR>
    <%= new java.util.Date() %>
  </BODY>
</HTML>
```

1.2.4 Výhody JSP

Oddělení statické části od dynamického obsahu má řadu výhod, dokonce i proti technologii servletů. Přístup, který se používá na stránkách JSP nabízí několik výhod i v porovnání s konkurenčními technologiemi, například ASP, PHP, ColdFusion, CGI, SSI atd.

Výhodou JSP proti ASP je, že dynamická část je napsaná v Javě, oproti VBScriptu nebo jinému jazyku určeného pro ASP. Java je komplexnější, robustnější a bezpečnější jazyk, který se hodí na tvorbu znovu použitelných komponent. Java je přenositelná, nejste limitovaní použitým operačním systémem anebo serverem [4].

Výhoda JSP proti PHP spočívá v tom, že JSP jsou psané v Javě, která je používaná pro psaní JSP a programátor se nemusí učit nový jazyk. Java poskytuje rozsáhlé třídy na práci se sítěmi, databázemi, elektronickou poštou atd.

Výhody JSP proti Servletům - JSP nedokáže něco, co by se nedalo v podstatě udělat i pomocí servletu. Ve skutečnosti to funguje tak, že stránka JSP se nejprve automaticky přeloží do servletu. Až tento servlet je potom vykonávaný na straně serveru.

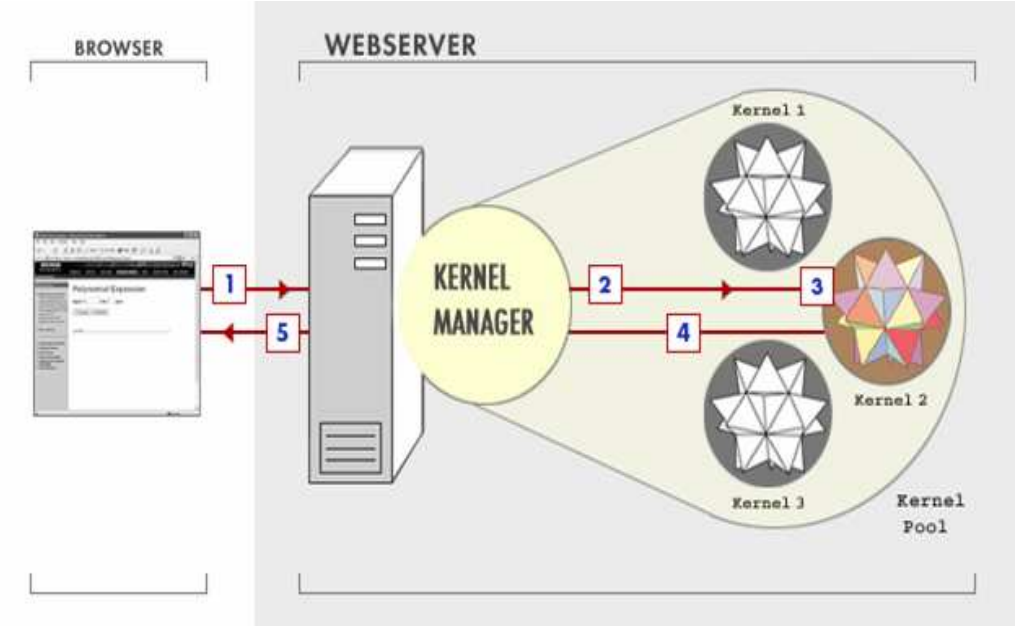
2 PROPOJENÍ PROSTŘEDÍ WEBMATHEMATICA S INTERNETEM

Základním problémem propojení je velikost textu v html stránce, protože matematické příkazy v programu Mathematica a upravené pro program webMathematica jsou dlouhé v poměru s ostatním textem psaným v html.

Problém s pomalým načítáním stránky kvůli její velikosti textu a zarovnáním výsledné stránky do internetového prohlížeče, zakomponování matematických zkratek a znaků do html kódu.

Ze začátku byla webMathematica a její stránky statické a pouze zobrazovaly text, ale po propojení webMathematica s dalšími internetovými službami (PHP, ASP a Java Scripts) se staly stránky webMathematica dynamické a uživatel může dokonale ovládat funkce a použití programu webMathematica.

Základem je vlastní kód pro www stránky a to Mathematica Server Pages (MSP), který je založen na technologii Java – servlety.



Obr. 6 – webMathematica technologie

Webový prohlížeč posílá požadavek na webMathematica server, webMathematica server si rezervuje Mathematica kernel (jádro) z bazénu, Mathematica kernel je nastaven se

vstupními parametry, provede kalkulace a vrací výsledky serveru, webMathematica server vrací Mathematica kernel do bazénu, webMathematica server vrací výsledky webovému prohlížeči.

3 DYNAMICKÉ WEBOVÉ STRÁNKY

Rozdíl mezi statickou a dynamickou webovou stránkou. Pod klasickou statickou stránkou se rozumí webová stránka, která není schopna měnit svůj vzhled poté, co je nahrána v prohlížeči.

Není tedy možné, aby se některé políčko tabulky např. zvýraznilo poté, co nad ním zastavíte kurzor, není možné, aby se na stránce odpočítával aktuální čas a rozhodně není v žádném případě povoleno, aby se vzhled jakýmkoli způsobem přizpůsoboval jednotlivým "serfařům".

Pokud by uživatel potřeboval jinou, byť pouze nepatrně modifikovanou stránku, bude třeba, aby byla ze serveru nahrána jiná stránka a půjde o dynamickou webovou stránku.

3.1 Dynamika na straně klienta

Tyto typické statické stránky bylo možno rozšířit o jistou interaktivitu začleněním ActiveX ovládacích prvků nebo Java appletů. Nicméně např. ActiveX prvky je nutné na počítač klienta nejdříve nahrát a zaregistrovat v registru, což je nepohodlné a často i nebezpečné, není-li zdroj těchto prvků spolehlivý. Proto došlo k vytvoření specifikace DHTML [4].

Za dynamické stránky lze považovat stránky, vyhovující specifikaci DHTML (Dynamic Hypertext Markup Language). Pod touto zkratkou se nerozumí žádný nový jazyk, jedná se o spolupráci klasického HTML se skripty, spouštěnými na straně klienta (nejčastěji JScript, méně pak VBScript), CSS (Cascading Style Sheets) a řady dalších technologií, které jsou společně propojeny objektovým modelem DOM (Document Object Model), schváleným W3C (World Wide Web Consorcium) 1.října 1998.

Pomocí DHTML lze dosáhnout efektů, které byly dříve téměř nebo zcela nemožné. Není již třeba implementovat ActiveX prvky (i když jsou někdy nepostradatelné), neboť alespoň částečnou podporu DHTML obsahují téměř všechny prohlížeče. JScript není obyčejně třeba ručně doinstalovávat a CSS podporují skoro všechny prohlížeče (do jisté míry).

Pomocí DHTML lze dosáhnout např. přizpůsobení vzhledu stránky jednotlivým uživatelům, působivých grafických efektů nebo např. implementace rozvíracích nabídek.

3.2 Dynamika na straně serveru

Pomineme-li dynamičnost webových stránek, kterou obstarává DHTML, lze pod dynamičností stránky rozumět rovněž schopnost přizpůsobit svůj vzhled parametrům, které získá od klienta, již na serveru, který stránku dynamicky podle těchto kritérií vygeneruje.

V praxi jde např. o situaci, kdy se člověk registruje do nějaké databáze a posléze obdrží stránku se zadanými parametry (třeba pro kontrolu a případnou změnu zadaných informací). Po shlédnutí zdrojového kódu bychom zjistili, že příslušná pole jsou běžně vyplněna např. jménem nebo příjmením registrovaného (nikoli skripty, které by identifikovaly pozici právě přidaného záznamu v databázi).

Rovněž je nesmysl, že by pro každého registrovaného byla připravena stránka nová, neboť nelze dopředu předpovědět, kdo se zaregistruje. V tomto případě mluvíme o skriptech, vykonávaných na straně serveru. Klientovi je vždy zaslán pouze výsledek těchto skriptů, nemá tedy možnost přistupovat k jejich zdrojovému kódu.

Pro interakci mezi serverovým skriptem a klientem je nutné nějak k serveru přenést uživatelem zadané informace. Tyto parametry jsou často součástí adresy webové stránky (např. http://www.neco.cz/stranka.asp?jmeno=Alena&prijmeni=Novakova).

Je však rovněž možné je umístit do hlavičky požadavku HTTP, takže zůstanou běžnému uživateli skryté.

Technologie, používané pro vygenerování stránek na serveru, lze rozdělit do dvou kategorií:

3.2.1 Skriptové vsuvky

V tomto případě se kód skriptu kombinuje přímo se zápisem webové stránky. Když je server vyzván k odeslání této stránky klientovi, nejprve ji celou projde, serverové skripty vykoná (jejich výstupem musí být HTML kód (resp. XML aj.)) a výsledek odešle klientovi.

Tato filozofie dynamicky generovaných stránek v současnosti převažuje. Dnes tímto způsobem fungují téměř všechny používané technologie (SSI, ASP, ASP.NET, PHP, JSP, ColdFusion).

3.2.2 CGI skripty

Tyto skripty jsou napsány v libovolném programovacím jazyce a zkompilovány do nějakého spustitelného souboru (např. exe). Pokud klient skript volá, volá vlastně spustitelný soubor, který od něj převezme parametry, na základě nichž vytvoří webovou stránku.

K psaní CGI-skriptů se nejčastěji používá jazyk Perl, nicméně lze bez problémů použít např. C/C++, Pascal nebo Javu.

3.3 CGI a ISAPI rozhraní

Webový server ve skutečnosti klientský požadavek nevykonává sám, nýbrž zavolá např. výše zmíněný CGI-skript nebo např. knihovnu ASP.DLL či modul ISAPI PHP. O vzájemnou komunikaci se pak musí starat rozhraní.

3.3.1 CGI (Common Gateway Interface)

Jak již název napovídá, CGI je rozhraní, které zajišťuje komunikaci mezi WWW-serverem a CGI-skriptem (=onen spustitelný soubor). Na CGI-skript jsou kladeny 2 požadavky, které musí splňovat: musí umět přebírat parametry, které obdrží od WWW-serveru, a výsledkem jeho činnosti musí být odpověď ve formátu HTTP (bez první stavové řádky).

CGI-skript lze napsat téměř v jakémkoli programovacím jazyce. Pokud je tímto jazykem Java, říká se těmto skriptům někdy servlety, což je analogie k termínu applet (= skript napsaný rovněž v Javě, vykonávaný však na straně klienta).

Komunikace tedy probíhá následovně: klient zašle požadavek WWW-serveru. Jakmile server zjistí, že adresa odkazuje CGI-skript, předá parametry skriptu a čeká na jeho vykonání.

Za chvíli obdrží výsledek ve formátu HTTP, který doplní o další hlavičky a odešle výsledek klientovi. Server mimochodem zjistí, jakou verzi HTTP klient používá a tomu přizpůsobí rovněž odpověď (např. nepošle hlavičky, jedná-li se o verzi 0.9).

CGI skripty mají poměrně velké pravomoci, tudíž je v zájmu administrátora, aby dovolil na server ukládat CGI-skripty pouze těmi lidmi, u nichž má zaručeno, že budou jejich programy bezpečné.

Většinou se CGI-skripty ukládají do speciálního adresáře (např. adresu typu http://server/cgi-bin/skript1.cgi již spatřilo jistě mnoho i laických surfařů).

Přestože technologie CGI ve své době rozpoutala horečku kolem dynamických stránek a napomohla růstu zájmu o Internet ze strany obchodního světa, má svá úskalí, která zapříčinila, že v současné době není u tvůrců dynamicky generovaných stránek oblíbena. Napsat CGI-skript je mnohem obtížnější než použít některý ze serverem vkládaných vsuvek (např. PHP nebo ASP).

Mimo to se při každém požadavku spouští pokaždé nová instance CGI-skriptu, což zabírá ohromné místo v paměti (na rozdíl např. od ISAPI, která se nahraje jednou do paměti a posléze slouží všem klientům; podporuje totiž více toků (threadů)).

3.3.2 ISAPI (Internet Server Application Programming Interface)

Aplikace ISAPI je rozhraním, které bylo vyvinuto firmou Microsoft a snaží se překonávat nejvíce omezující prvky staršího CGI. ISAPI se spoléhá na dynamicky připojované knihovny (DLL).

Aplikace ISAPI je tvořena jednou knihovnou DLL, která se nahrává do stejného paměťového prostoru jako webový server v okamžiku, kdy je poprvé požadována. Následně je pro obsluhu všech dalších klientů používána právě tato instance, čímž odpadá zatěžování serveru nahráváním stále nových instancí jako u CGI.

Všechny knihovny DLL ISAPI musí umožňovat zavedení více toků, aniž by to způsobovalo problémy s funkcemi aplikace.

Technologie ISAPI rovněž dovoluje vyvíjet filtry. Filtr ISAPI je zvláštní knihovna DLL, která se nachází ve stejném paměťovém prostoru jako webový server. Webový server ji volá jako reakci na každý požadavek HTTP. Filtr ISAPI tedy instruuje server, jak má požadavek klienta zpracovávat.

3.4 Jednotilivé technologie pro generování stránek na serveru

3.4.1 ASP (Aktive Server Pages) a ASP.net

ASP je v podstatě filtr ISAPI. Tato technologie byla uvedena roku 1996 firmou Microsoft. Její kódové jméno bylo Denali. ASP je uložena v jediné knihovně s názvem ASP.DLL (asi 300 KB).

Kdykoli klient požaduje stránku .asp, předá webový server jeho žádost filtru ISAPI ASP, který se postará o vykonání všech skriptů (k tomu musí nahrát knihovnu s daným skriptovacím jazykem). Výsledek ASP je kód HTML (nebo XML aj.), jenž se vloží do textového toku, odeslaného klientovi.

Součástí knihovny ASP.DLL je objektový model, který vývojáři ASP stránek umožňuje různé akce. Základními sedmi objekty ASP verze 3.0 jsou Application, ASPError, ObjectContext, Request, Response, Server a Session. Application reprezentuje samostatnou ASP aplikaci, Request slouží k manipulaci s příchozími informacemi od klienta (cookies, hlavičky, certifikační informace, informace z formulářů aj.),

Response přistupuje k HTTP reakci odesílané klientovi (např. metoda Write zapisuje na stránku textový řetězec), Server dovoluje přistupovat k webovému serveru (a přistupovat např. k zavedeným objektům), Session obsahuje informace o každém aktivním spojení mezi serverem a klientem, ASPError slouží k ošetřování chyb a ObjectContext umožňuje vytvořit transakční aktivní serverové stránky (určitý blok skriptu se buď vykoná celý nebo vůbec). Poslední dva zmíněné objekty jsou novinkami ve verzi 3.0.

ASP je v současné době spolu s PHP nejrozšířenější technologií v oblasti aplikací pro dynamickou tvorbu webových stránek. Oproti PHP má však řadu nevýhod: je vázána na operační systém Microsoft (PHP lze bez problémů použít pod Linuxem), je pomalejší než PHP, zprovoznění serveru, který by ASP provozoval, je složitější a nákladnější a ASP není ani tak bezpečná jako PHP.

Chceme-li si zprovoznit server, schopný obsluhovat ASP stránky, je nutné na něj nainstalovat buď program IIS (Internet Information Server, nyní ve verzi 5.0), který funguje pouze pod Windows NT/2000/XP nebo jeho jednodušší verzi PWS (Personal Web Server), fungující i pod Windows 95/98. PWS se hodí pro testování, neboť podporuje téměř

všechny funkce jako IIS. Nicméně dovoluje současnou obsluhu maximálně desíti uživatelů, což jeho použití omezuje maximálně na malý intranet.

Druhou možností, jak publikovat web založený na ASP technologii, je jeho nahrání na server některého poskytovatele. Počet free-hostingových služeb, které by ASP podporovaly, je pouze minimální (na rozdíl od PHP). Pro profesionálnější účely tedy člověku nezbude nic jiného, než si sám zřídit webový server nebo využít některou z placených služeb.

Po uvedení komplexní síťové platformy .NET Microsoftem v únoru 2002 byla rovněž představena nová verze ASP. ASP.NET se od tradičního ASP (nyní nazývaného Classic ASP) velice liší. Viditelný rozdíl i pro běžného uživatele bude přípona těchto stránek - .aspx na rozdíl od dosavadní .asp.

Tyto stránky mohou být napsány v mnoha jazycích (C#, Visual Basic, C++, J# aj.). Při prvním volání je stránka převedena do jazyka MSIL (Intermediate Language). Tento kód je následně zkompilován do nativního kódu, který je posléze spuštěn. Jeho výsledkem je pak HTML kód, který putuje přes webový server ke klientovi.

Při všech následných dotazech na tutéž stránku se již ale přistupuje přímo k onomu zkompilovanému kódu, čímž se zvýší rychlost reakce serveru na klientův požadavek.

ASP.NET tedy přináší nový přístup. Nicméně stále zůstává fixována na platformu systému Windows. Výhodou je alespoň možnost bezplatné možnosti downloadu platformy .NET.

Classic ASP lze programovat např. v MS Visual InterDev (součást balíku Microsoft Visual Studio) a pro ASP.NET slouží MS Visual .NET Studio.

3.4.2 SSI (Server Side Include)

Jde o nejstarší a nejrozšířenější druh vkládaných vsuvek. Použití je zcela jednoduché: do stránky stačí přidat zápis ve tvaru <!--#příkaz parametr="hodnota"--> a server tuto značku poznámky nahradí za příslušný textový řetězec, podporuje-li SSI. V opačném případě se řetězec nevyhodnotí a klientovi je odeslána poznámka, která se samozřejmě nezobrazí.

Na stránce tak můžeme zobrazit např. velikost příslušného dokumentu, datum jeho poslední modifikace, datum a čas na serveru apod.

Následující příklad demonstruje použití příkazu #echo, který vloží na stránku datum a čas poslední úpravy souboru.

3.4.3 PHP (Hypertext Preprocesor)

PHP je skriptovací jazyk, který se přímo začleňuje do textu HTML. Historie PHP sahá do roku 1994, kdy se jistý Rasmus Lerdorf rozhodl naprogramovat v Perlu množinu skriptů sloužící k evidenci přístupu k jeho stránkám. Později Lerdorf systém importoval do jazyka C. Nedlouho po uvolnění úvodní verze PHP začalo PHP používat stále více lidí. Zanedlouho Lerdorf přidal ještě program Form Interpreter (FI), který zpřístupňoval databázové systémy na webu. Nový systém, nyní pod názvem PHP/FI, zaznamenal celosvětovou popularitu.

V současné době je PHP doplněno o množství dalších funkcí a nejnovější verzí je verze 4.2.2.

Funkce PHP se velmi podobá ASP. Pokud webový server zjistí, že požadavek HTTP směřuje na PHP stránku, je dotaz přesměrován do modulu PHP ISAPI, který celou stránku zkompiluje. Poté je výsledek předán zpět webovému serveru, který jej odešle klientovi. Nicméně na systému PHP je zajímavé, že může pracovat i jiným způsobem: lze jej spustit jako modul serveru Apache (pouze na Unixu) nebo může pracovat jako CGI-skript.

PHP za svou popularitu vděčí nezávislosti na platformě (lze jej přenášet mezi Windows, Unix, Mac OS, OS/2), jednoduchosti syntaxe a stabilnosti. Je šířen bezplatně. Lze jej provozovat např. na serverech Apache, TomCat nebo MS IIS. K vývoji PHP webu lze použít např. programy Macromedia Dreamweaver MX, NuSphere PHP nebo EdZend Studio.

3.4.4 JSP (Java Server Pages) a J2EE (Java 2 Enterprise Edition)

Roku 1997 byly firmou Sun Microsystems představeny malé programy, generující webové stránky na straně serveru - servlety. O rok později Sun představuje komplexní platformu J2EE, na jejímž základě jsou postaveny i produkty jiných softwarových společností.

Roku 1999 se Sun nechal inspirovat technologií ASP a uvádí specifikaci JSP, které je postavena platformě J2EE. JSP má oproti servletům tu výhodu, že pro její programátory

jednodušší. Není totiž nezbytné programovat celý servlet, ale pouze dodat do HTML kódu několik značek a efekt je nakonec stejný.

Při prvním volání JSP stránky od její změny je stránka nejprve konvertována do podoby servletu, pak zkompilována do bytového kódu (.class) a posléze pomocí rozhraní JVM spuštěna. Výsledek je pak odeslán webovému serveru, který výslednou HTML (popř. XML aj.) stránku pošle klientovi.

Při dalším požadavku klienta již není stránka převáděna do bytového kódu. Odpadá tedy konverze na servlet a bytový kód. Stránka je tedy podruhé mnohem rychleji nahrána.

Technologie J2EE je nezávislá na hardwarové platformě a operačním systému. Jde o komplexní a nákladné řešení, které se hodí pro vývoj velkých internetových a intranetových aplikací.

II. PRAKTICKÁ ČÁST

4 WEBOVÉ ROZHRANÍ PRO EVOLUČNÍ ALGORITMY

Navržené internetové rozhraní pro evoluční algoritmy jsem psal v jazyku html. Stránky jsou optimalizovány na jednoduchost a rychlost. Základní rozhraní je rozděleno na český a anglický jazyk. Zdrojový kód index.htm je v příloha P III.

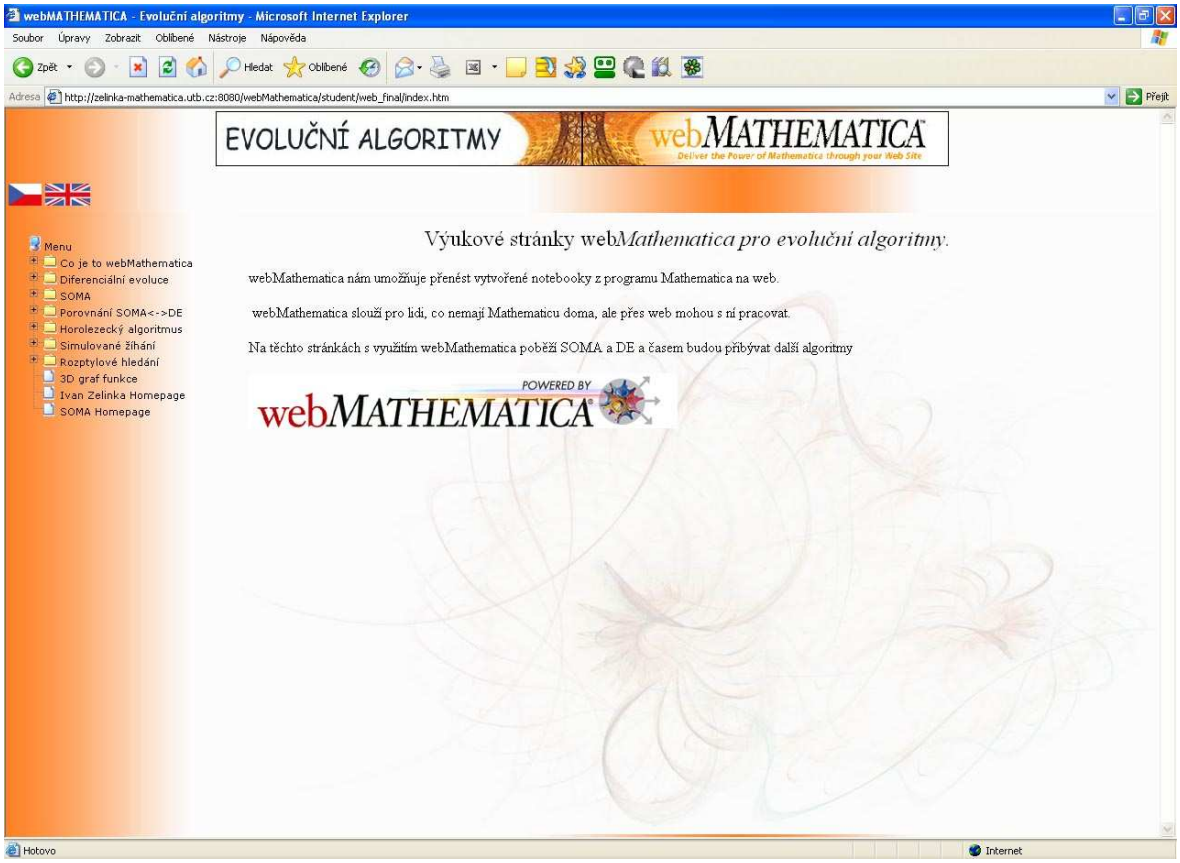
Pak je stránka rozdělena na levý sloupec menu a zobrazovací část, kde se budou otevírat nové stránky. Hlavní iframe webu evolučních algoritmů má název main a nově otevřená okna se v něm budou zobrazovat.

Levé sloupcové menu je naprogramováno v jazyku Java a je zobrazeno ve stromovém tvaru. Po kliknutí na menu se vždy zobrazí popis námi zvoleného tématu a možnost vyzkoušet si funkční příklad evolučního algoritmu, který poběží v programu webMathematica.

Hlavní odkaz na evoluční algoritmy v prostředí webMathematica je

http://zelinka-mathematica.utb.cz:8080/webMathematica/evoluce/

Obr. 7 ukazuje úvodní stránku, která se Vám zobrazí po zadání odkazu do www prohlížeče.



Obr. 7 – Internetové rozhraní

4.1 Rozdělení stránek

Pro jednoduchost ovládání stránky bylo vytvořeno stromové menu, které se může jednoduše modifikovat. Přidávat a nebo odebírat staré a nové stránky na webu pomocí úpravy jedné html stránky.

Stromové menu je vlevo v sloupečku. Odkazy ve stromovém menu můžou odkazovat na novou stránku, otevření nového okna internetového prohlížeče a nebo otevírat stránky do iframe main.

4.2 Modifikace některých funkcí

Protože webMathematica, ale i Mathematica využívá stejných funkcí, které jsou i v jazyku html, musely se proto některé funkce při psaní *.jsp souboru v programu webMathematica

upravit tak, aby nevyužívali stejného názvu jako html a byla možná funkčnost www stránek, ale i webMathematica.

Pro příklad uvádím část kódu v Mathematica *.nb a pod ta samá část ale psaná webMathematica *.jsp

ukázka Mathematica nb

….

```
Population = DoPopulation[PopSize, Specimen];
HowManySimulations = 10;
AllFinalPopulation =Table[NestList[SOMAATO, Population, Migrations],
{i,HowManySimulations}];
```

….

ukázka webMathematica jsp

….

```
<msp:evaluate>
Population=DoPopulation[PopSize,Specimen];
If[MSPValueQ[$$varMigrations],HTMLTableForm[Transpose[Population]]]
</msp:evaluate>
<br>
Finální populace
<br>
<msp:evaluate>
FinalPopulation = NestList[SOMAATO, Population,Migrations];
If[MSPValueQ[$$varPopSize], HTMLTableForm[Transpose[Last[FinalPopulation]]]]
</msp:evaluate>
```

….

Jazyk psaný v webMathematica sám nabízí ochranu proti kopírování a to pomocí v sobě funkčního modulu MSP, který je obsažen ve webMathematica a tím nemůže dojít ke kopírování stránky s obsahem zdrojového kódu.

MSP má ještě funkci, která zajišťuje optimální chod web Mathematica při připojení více uživatelů najednou a tím nemůže dojít k pádu nebo přetížení systému a jeho následné nefunkčnosti.

Pokud spadne server webMathematica, www stránka zobrazuje chybu serveru, protože nejede jádro programu webMathematica.

4.3 Použité funkce webMathematica pro evoluční algoritmy

MSP [3] funkce pro vkládání do *.jsp souboru ve webMathematica jsou rozděleny podle použití a potřeby naprogramování stránek. Pro evoluční algoritmy SOMA a DE jsem použil funkce :

4.3.1 MSP

MSP je rozděleno do šesti funkcí, které jsou popsány v následující Tab.1

Tab. 1 Šest funkcí MSP

msp:allocateKernel	přiděluje Matematice jádro pro počítání
msp:evaluace	ohodnotí vstup do Mathematica a vloží výsledek do výstupní schránky
msp:get	dostane výsledek z Mathematica výpočtu a použije ho pro nastavení Java výrazu
msp:set	nastaví Mathematica proměnnou s hodnotou Java výrazu
msp:includeClassicMSP	obsahuje klasický MSP script
msp:forwardClassicMSP	přednastaví klasický MSP script

Většina základních použití MSP tagů je pro JSP zabalený v msp:allocateKernel tag okolo jednoho a nebo více msp:evaluate tagů.

Ukázka:

```
....
<msp:allocateKernel>

<msp:evaluate>
    ...
</msp:evaluate>

<msp:evaluate>
...
</msp:evaluate>

</msp:allocateKernel>
...
```

Následující popis je pro operace s tagy. Tohle poskytne přehled kroků v zpracování JSP, které používá MSP taglib.

msp:allocateKernel (tag open)

msp:allocateKernel tag je užíván pro získání Mathematica jádra pro počítání. Otevře tag a bere následující kroky pro předzpracování stránky.

Determine the Pool

msp:allocateKernel poprvé určí bazén, který je založený na žádosti na jméně JSP. Jestli není žádný bazén pojmenován, je využíván všeobecný.

Allocate the Kernel

Mathematica jádro je požadováno z jádra bazénu. Bazén udržuje sbírku Mathematica jáder čekajících na výpočty. Jestliže není žádné jádro k dispozici, systém čeká až se některý uvolní a je připraven. Používání bazénu dovolí systému sdílet Mathematica jádra přes vícenásobné žádosti, které vedou k rychlejšímu času reakce pro systém. Každá žádost může dostat jiné jádro! Vy se tedy nemůžete spolehnout na uložení Vašeho jádra Mathematica a jeho znovuobnovení.

4.3.2 SetSecurity[];

SetSecurity je důležité pro nastavení bezpečnostního obsahu stránky, znemožňuje kopírovat obsah stránky se zdrojovým kódem. Internetové zabezpečení se týká jak klienta tak i serverové bezpečnosti. Nicméně, webMathematica je serverová technologie. Následkem toho, tam nejsou žádné zvláštní klientské bezpečnostní problémy vztahující se k webMathematica JSP. Samozřejmě, vy ještě potřebujete uvážit klientovskou-postranní bezpečnost, ale Mathematica nepředloží žádné zvláštní klientské bezpečnostní problémy.

Běh hlavního výpočetního systému Mathematica ve webové stránce představuje mnoho potenciální ochrany hazardu pro server. Mathematica obsahuje příkazy pro náhled a vymazání souborů a pro vypínání libovolných procesů. Několik bezpečnostních rysů je postaveno pro nástroje webMathematica, ale tyto jsou navržené k tomu, aby pracovaly spolu s dalšími standardními bezpečnostními rysy.

4.3.3 Needs["MSP`HTML`"]

Jedna z výhod HTML šablony pro webMathematica je, že poskytne možnost programování HTML formátu přímo ve Mathematica programech. webMathematica podporuje funkce pro tabulkový výpis. Jestli chcete vrátit HTML, které není vygenerované HTML balíkem, měl by konstruovat Váš vlastní řetěz.

The HTML Functions

The HTML funkce jsou obsaženy v balíku, MSP`HTML`, který je součástí webMathematica rozvržení.

HTMLTableForm

Jak je vysvětleno výše, MSP`HTML` balík je k dispozici pro webMathematica a může být nainstalovaný do regulárního prostředí Mathematica.

5 UKÁZKOVÉ EVOLUČNÍ ALGORITMY V PROSTŘEDÍ WEBMATHEMATICA

Evoluční algoritmy, které poběží na serveru webMathematica je SamoOrganizující se Migrační Algoritmus (SOMA) a Diferenciální Evoluce (DE). Je to skupina evolučních algoritmů, která je popsána v [2]. Základní struktura SOMA a DE byla převzata z notebooku, který je přidán v příloze k diplomové práci a v prostředí Mathematica je plně funkční.

Tenhle notebook jsem zkompiloval do *.jsp stránky se základními funkcemi pro webMathematica a názorná ukázka je na internetovém rozhraní, které jsem vytvořil.

Některé parametry funkcí musely být upraveny, protože se shodovaly s parametry pro psaní v jazyku html nebo jsp. Základem stránky je html jazyk, do kterého jsou přidány funkce webMathematica.

Stejným způsobem jakým byla napsána SOMA do *.jsp byla napsána i DE. V dalším příkladě je zkouška a vytvoření stránky, která vypíše a vyhodnotí oba dva algoritmy najednou a výsledkem jsou tabulky a grafy obou dvou funkcí v přímém srovnání.

5.1 Účelové funkce

Pro oba dva algoritmy se musí zadávat vstupní účelová funkce, která se bude optimalizovat.

V nabídkovém rozbalovacím menu je připraveno 16 testovacích funkcí De Jong 1st, De Jong 2nd, De Jong 3rd, De Jong 4th, Rastrigin, Schwefel, Griewangk, Sine Wave, Stretched Sine, Test Function Ackley, Ackley, Egg Holder, Rana, Pathological, Michalewicz, Masters [2].

Všechny tyto funkce byly testovány v programu Mathematica a webMathematica a jejich výsledky byly totožné v obou programech a tím byla ověřena správná funkčnost napsaných evolučních algoritmů v webMathematica.

Pokud uživatel nezvolí jednu z 16-ti testovacích funkcí, zadává vlastní funkci, kterou bude zadaný algoritmus počítat.

5.2 SOMA

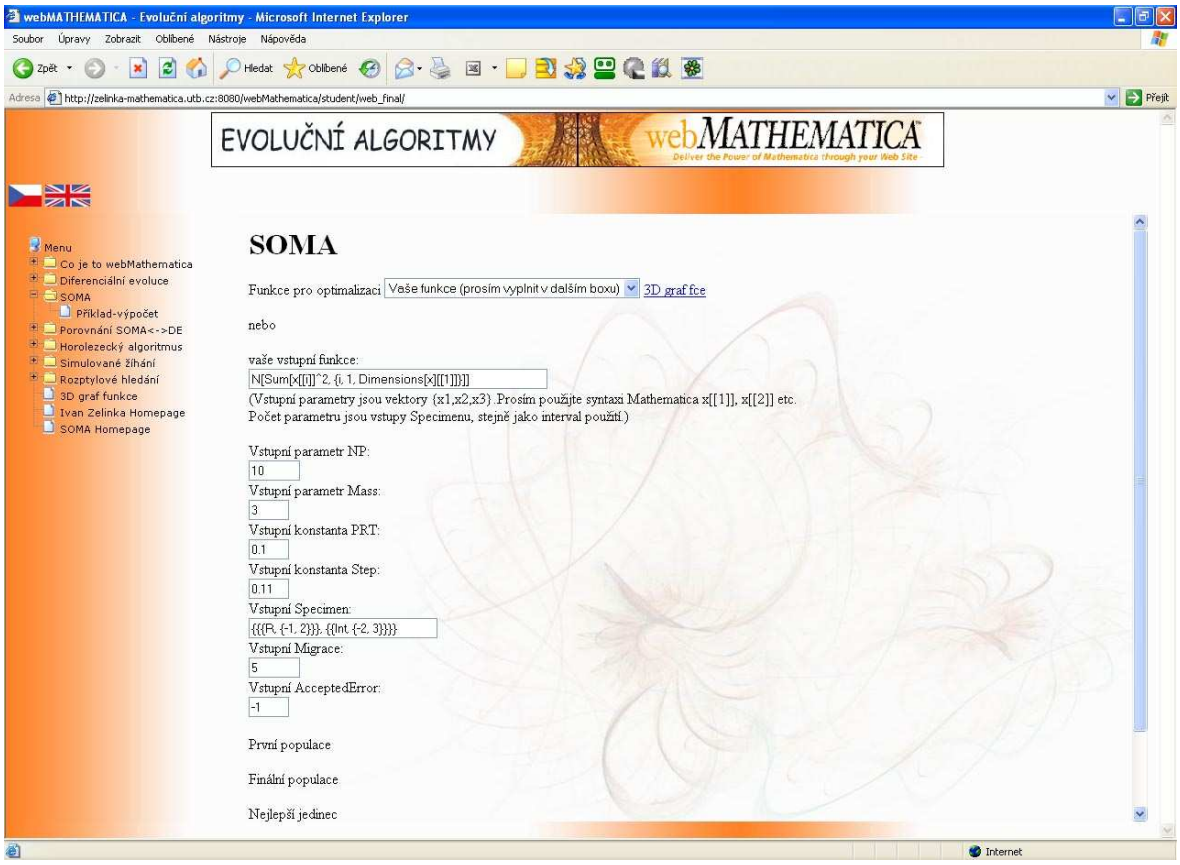
SOMA je algoritmus existující od roku 1999, jehož činnost je založena na geometrických principech.Vzhledem k tomu že spolupracuje s populace podobně jako genetické algoritmy a výsledek po jednom evolučním cyklu (migračním kole) je totožný s genetickými algoritmy či diferenciální evolucí, lze jej řadit například mezi evoluční algoritmy navzdory faktu, že během jeho běhu nejsou vytvářeni noví potomci, jak je to u jiných evolučních algoritmů. Původní myšlenka, která vedla k vytvoření, spočívá v napodobení chování skupiny inteligentních jedinců, kteří kooperují při řešení společného problému jako je např. hledání potravy apod. SOMA doznala spousty změn, až do této doby kdy robustní ve smyslu nalezení globálního extrému a vyrovnává se takovým algoritmům jako je diferenciální evoluce. SOMA je založena na kooperativním prohledávání (migraci) prostoru možných řešení daného problému, řídí se principy vycházejícími ze spolupráce inteligentních jedinců migrujících v prostoru možných řešení, je pro evoluční cyklus známý jako „Generace“ zvolen název „Migrační kolo“. Vlastnost samoorganizace u SOMA algoritmu plyne z faktu, že se jedinci ovlivňují navzájem během hledání lepšího řešení, což mnohdy vede k tomu, že v prostoru možných řešení vznikají skupiny jedinců, které se rozpadají či spojují, putují přes prohledávaný prostor. Jinými slovy si skupina jedinců neboli populace sama organizuje vzájemný pohyb jedinců – odtud samoorganizace.

Základem SOMA algoritmu je vybrání si funkce, kterou budeme optimalizovat a nebo vložení si vlastní funkce, kterou budeme řešit. V úvodu je popsán princip SOMA algoritmu jeho počátečních parametrů a vlastně co ta funkce provádí a jak funguje.

Pak se musí zadat vstupní parametry do algoritmu. Zadané hodnoty musí být v intervalu přístupných pro tento algoritmus, pokud se zadá větší číslo, výpočet se neprovede a musí se zadat odpovídající hodnota. Vstupní veličiny jsou chráněny proti tomu, aby uživatel zadal nepovolený znak a algoritmus mohl fungovat. Práce s www stránkou SOMA je jednoduchá a uživateli zobrazí výsledek nejlepších jedinců v tabulce a pak následný graf nejlepších jedinců celé populace v průběhu evoluce.

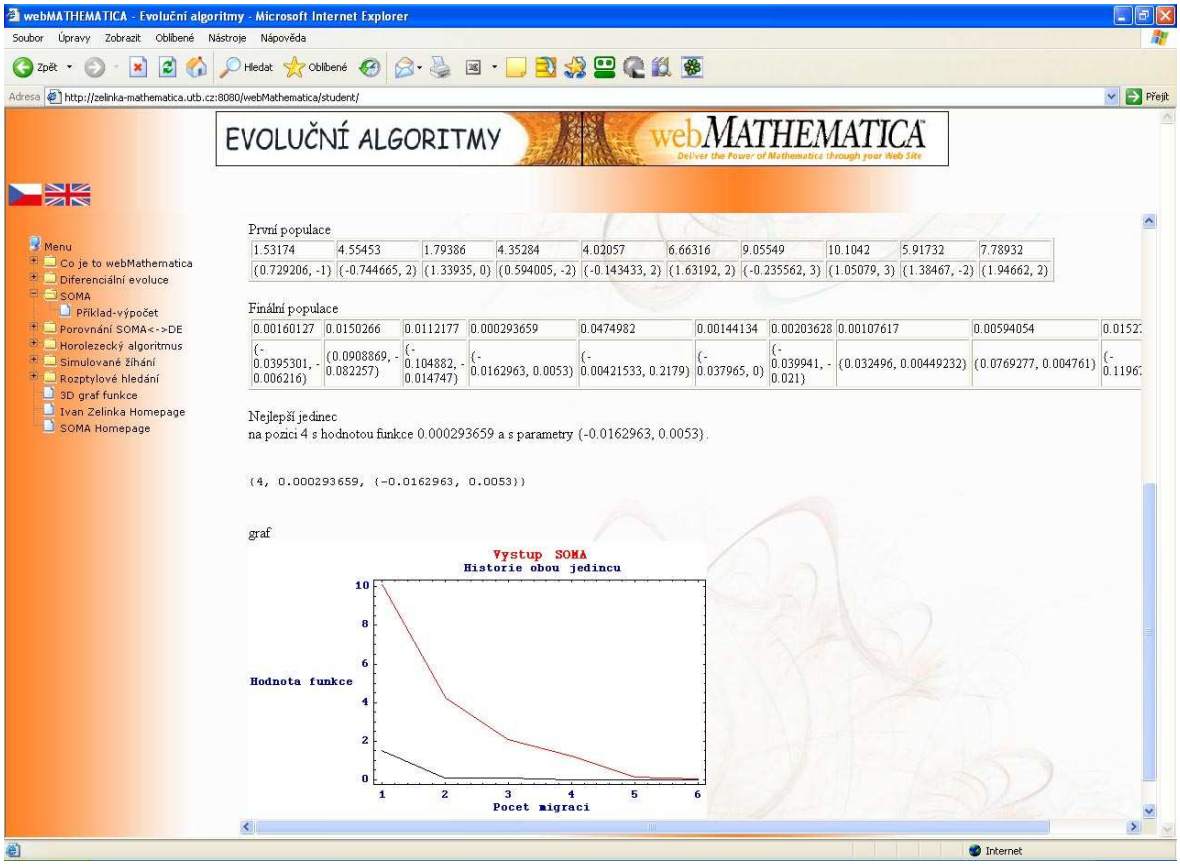
5.3 Příklad SOMA

Na obr.8 je ukázka zobrazení SOMA algoritmu v internetovém prohlížeči s možností zadávání vstupních dat a následného výpočtu. Zdrojový kód soma.jsp je v příloha P I.



Obr. 8 – Zadávání parametrů SOMA

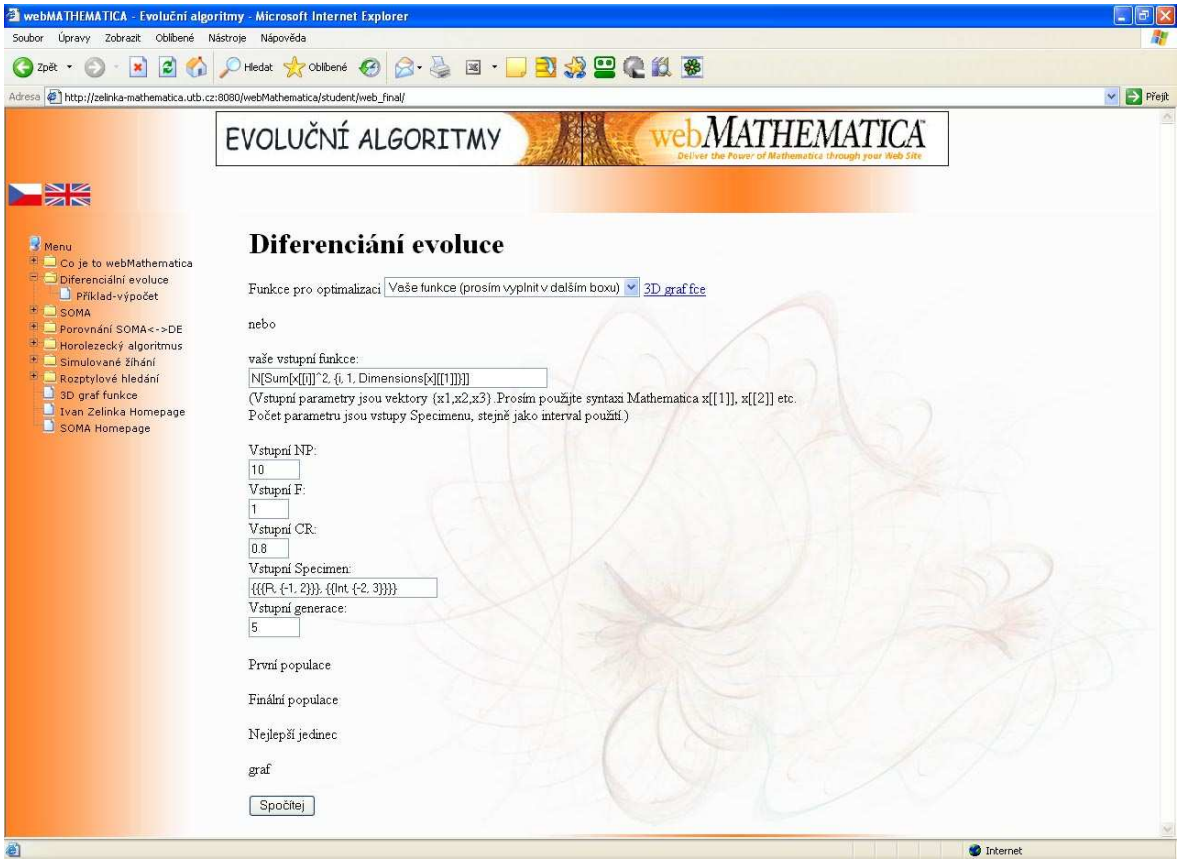
Obr.9 ukazuje výsledky výpočtu algoritmu a tabulkové zobrazení první populace, finální populace, vypsání nejlepšího jedince populace a graf.



Obr. 9 – Ukázka výpočtu SOMA

5.4 Diferenciální evoluce

Diferenciální evoluce (DE) je poměrně nový typ evolučního algoritmu (od r. 1995), který vyvinuli a poprvé použili Ken Price a Rainer Storm. Jeho schéma je dost podobné algoritmům genetickým, s nimiž má několik společných rysů, jako je například tvorba potomků (zde však pomocí 4 rodičů), používání tzv. generací apod. R.Storm navrhl vytváření populace potomků ve zvláštní populaci, jenž se účastí soupeření o místo v nové populaci až po naplnění této zvláštní populace [2]. Cílem diferenciální evoluce je v cyklech zvaných „generace“ vyšlechtit co nejlepší populaci jedinců ve smyslu hodnot účelové funkce, jenž je spojena s každým jedincem. Zdrojový kód de.jsp je v příloha P II.

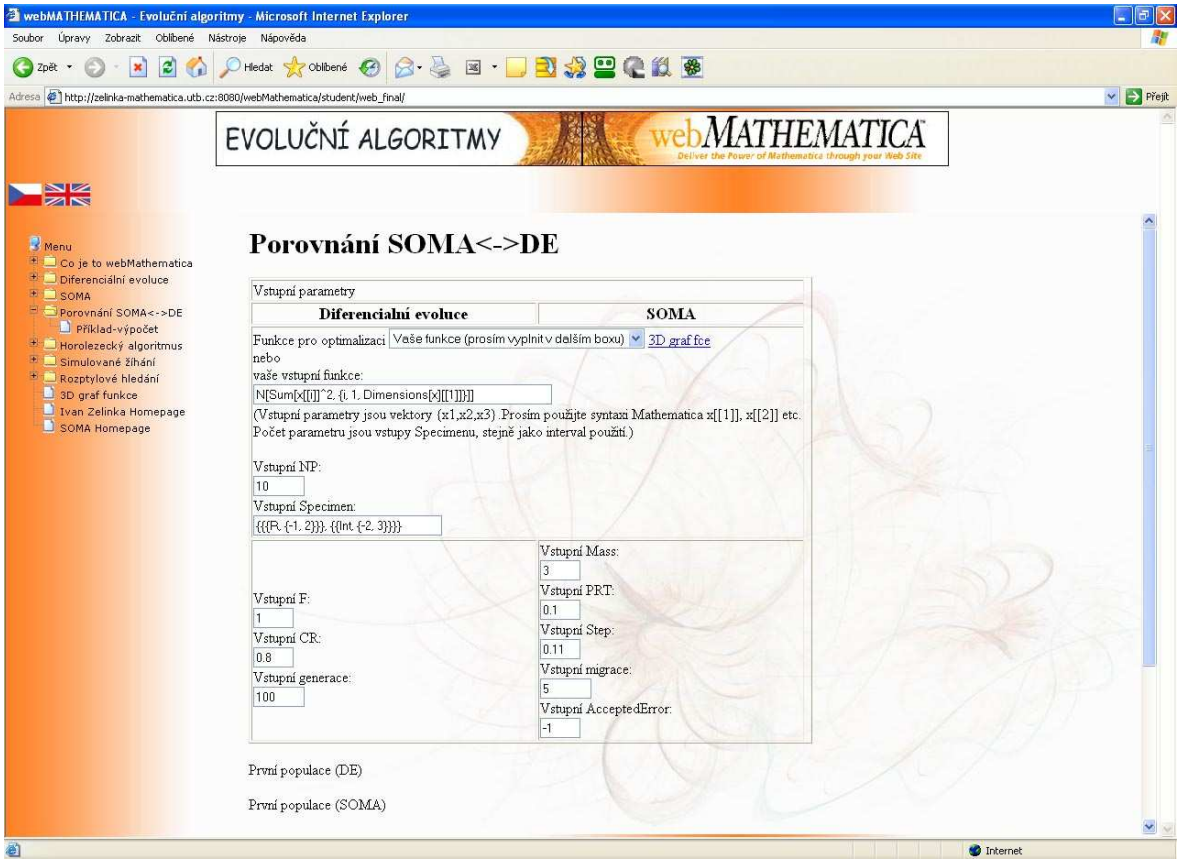


Obr. 10 – Ukázka www Diferenciální evoluce

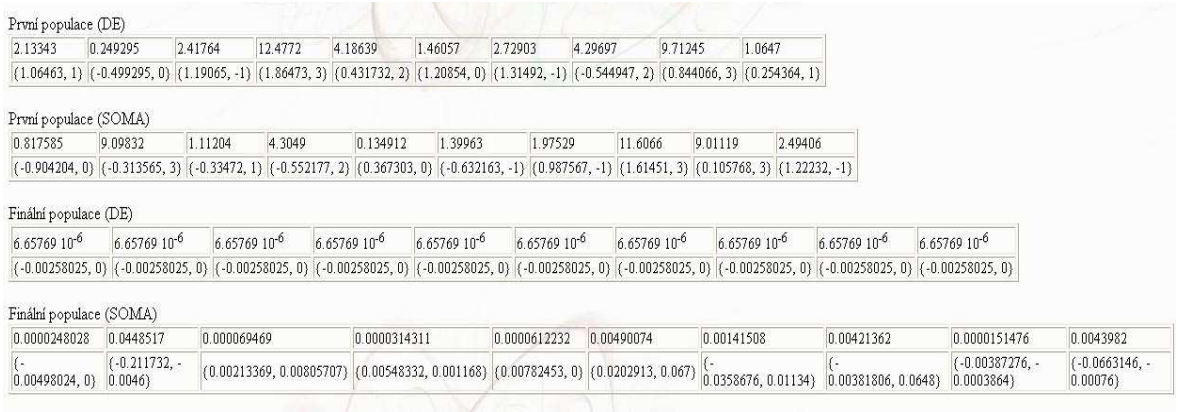
5.5 Porovnání SOMA a DE

Vytvořená *.jsp stránka počítá dohromady s oběma evolučními algoritmy a výsledkem je zobrazení výstupních tabulek, hodnot a grafů v jedné stránce a tím umožněno přímé porovnání algoritmů.

Pro zadávání stejných vstupních parametrů v SOMA a DE se použila jedna vstupní hodnota pro oba dva algoritmy, pro další parametry byla tabulka rozdělena na zadávání vstupních parametrů zvlášť pro SOMA a DE.



Obr. 11 – Ukázka www Porovnání SOMA <-> DE



První populace (DE)

2.13343	0.249295	2.41764	12.4772	4.18639	1.46057	2.72903	4.29697	9.71245	1.0647
{1.06463, 1}	{-0.499295, 0}	{1.19065, -1}	{1.86473, 3}	{0.431732, 2}	{1.20854, 0}	{1.31492, -1}	{-0.544947, 2}	{0.844066, 3}	{0.254364, 1}

První populace (SOMA)

0.817585	9.09832	1.11204	4.3049	0.134912	1.39963	1.97529	11.6066	9.01119	2.49406
{-0.904204, 0}	{-0.313565, 3}	{-0.33472, 1}	{-0.552177, 2}	{0.367303, 0}	{-0.632163, -1}	{0.987567, -1}	{1.61451, 3}	{0.105768, 3}	{1.22232, -1}

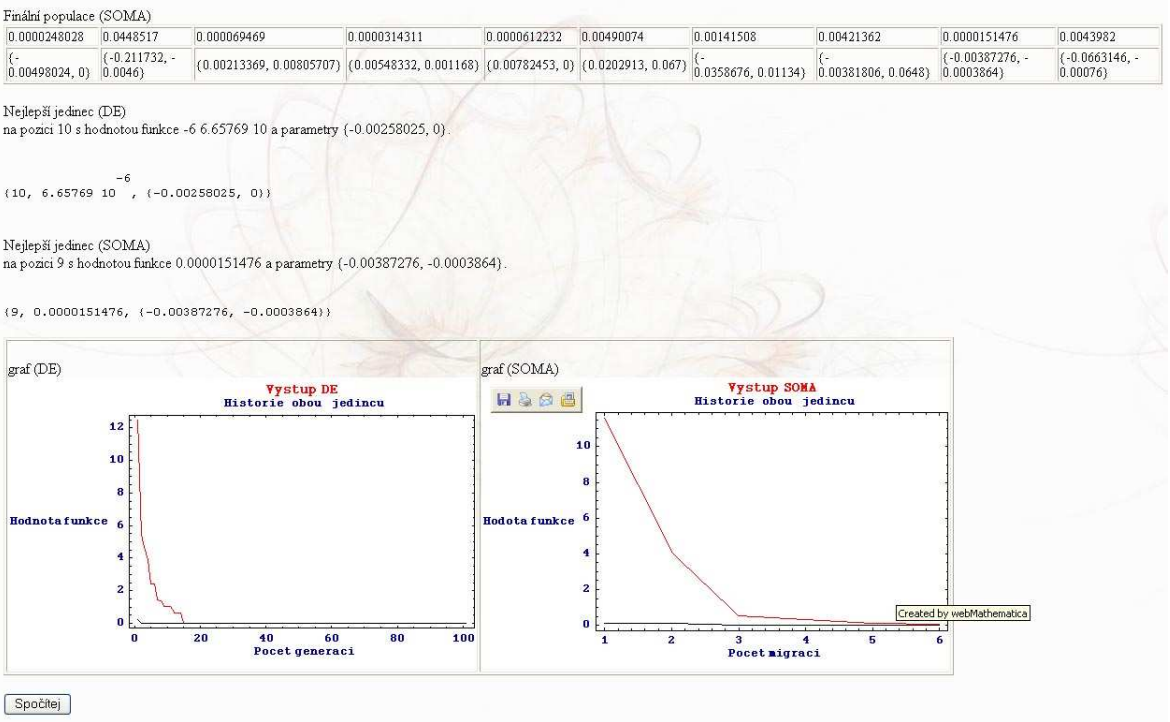
Finální populace (DE)

$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$	$6.65769\ 10^{-6}$
{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}	{-0.00258025, 0}

Finální populace (SOMA)

0.0000248028	0.0448517	0.000069469	0.0000314311	0.0000612232	0.00490074	0.00141508	0.00421362	0.0000151476	0.0043982
{-0.00498024, 0}	{-0.211732, -0.0046}	{0.00213369, 0.00805707}	{0.00548332, 0.001168}	{0.00782453, 0}	{0.0202913, 0.067}	{-0.0358676, 0.01134}	{-0.00381806, 0.0648}	{-0.00387276, -0.0003864}	{-0.0663146, -0.00076}

Obr. 12 – Výstupní tabulky SOMA a DE



Obr. 13 – Výstupní hodnoty a grafy SOMA a DE

ZÁVĚR

První část diplomové práce je zaměřena na teorii, která popisuje software Mathematica a webMathematica. Dále jsou zde zpracovány základní pojmy z oblasti dynamických internetových stránek a technologie pro generování stránek na straně klienta a serveru.

Druhá část diplomové práce popisuje vytvoření funkčního internetového rozhraní pro evoluční algoritmy pod softwarovým prostředím webMathematica. Použití funkcí podporované softwarem webMathematica je zde podrobně vysvětleno.

S využitím vytvořeného internetového rozhraní byly vyzkoušeny dva evoluční algoritmy a to - Samo-Organizující Se Migrační Algoritmus (SOMA) a Diferenciální Evoluce (DE) s možností přímé ukázky jejich výpočtu v prostředí webMathematica. Oba dva algoritmy byly naprogramovány pro softwareový program webMathematica samostatně, ale také je zde přístupné společné porovnání výpočtu obou algoritmů. V současné verzi bylo porovnávání SOMA a DE naprogramováno do jednoho *.jsp souboru. V budoucím rozšíření stránek o další algoritmy se předpokládá vytvoření jednoho *.jps souboru, který by načítal funkce s evolučními algoritmy a byla by možnost výběru jakýchkoliv dvou či více algoritmů a jejich porovnání na stránkách webMathematica. Oba dva evoluční algoritmy byly testovány na účelových funkcích v Mathematica i webMathematica, aby se prokázalo správné naprogramování těchto evolučních algoritmů pro internetové rozhraní s programem webMathematica. Chování bylo v obou verzích totožné.

Internetové rozhraní webMathematica je tedy v současné době plně funkční pro evoluční algoritmy na stránce http://zelinka-mathematica.utb.cz:8080/webMathematica/evoluce/

s možností plynule přidávat další algoritmy do internetového rozhraní. V budoucnu by tyto stránky měly být informačním bodem o evolučních algoritmech, které budou do stránek postupně přidávány.

ZÁVĚR V ANGLIČTINĚ

First part of master thesis is sight on theory, which describes software Mathematica and webMathematica. Further there are processed basic concepts from areas of dynamic internet pages and technology for generating pages on the side client and server.

Second part of master thesis describes creation of functional Internet interface for evolutionary algorithms under the software environment webMathematica. Using of function supported software webMathematica is here in detail explication.

With created web interface two evolutionary algorithms were tested, namely - the Self-Organizing Migrating Algorithm (SOMA) and Differential evolution (DE) with possibility of direct explanation of their calculation in the environment webMathematica. Both algorithms were programmed for software webMathematica separately, but also a comparison of calculations of both algorithms are accessible here. In current version, the collation of SOMA and DE was programmed into a one main *.jsp file. In future, it is suppossed an enlargement of pages for other algorithms presuming to create one *.jps file, that would load functions with evolutionary algorithms and it would have possibility of selection of any two or more algorithms and their comparison on pages webMathematica together. Both evolutionary algorithms were testing on benchmark functions in Mathematica and also webMathematica, to verify correct function of these evolutionary algorithms for internet interface with programme webMathematica. Behaviour was in both version identical.

Internet interface webMathematica is then, at present, in fully functional version for evolutional algorithms on url address

http://zelinka-mathematica.utb.cz:8080/webMathematica/evoluce/

with possibility fluently addition of other algorithms to this interface. In future, these pages should be an informative point about evolutionary algorithms, that will be added into the pages step by step.

SEZNAM POUŽITÉ LITERATURY

[1] Kvasnička V., Pospíchal J., Tiňo P., Evolučné algoritmy, STU Bralislava, 2000, ISBN 80-227-1377-5

[2] Zelinka I., Umělá inteligence v problémech globální optimalizace, BEN, 2002, 190 p. ISBN 80-7300-069-5

[3] Wickham – Jones T., webMathematica: A User guide, version 2.2., březen 2005

[4] Flanagan D., JavaScript Kompletní průvodce, 2. aktualizované vydání, Computer Press, 2002, Praha, ISBN: 8072266268

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASP	je technologie nezávislá na programovacím jazyce, která umožňuje vykonávání kódu na straně serveru a následné odeslání výsledku uživateli.
ASP.NET	je nadstavba .NET Frameworku firmy Microsoft pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages).
Apache	je softwarový webový server s otevřeným kódem pro Linux, BSD, Microsoft Windows a další platformy. V současné době dodává prohlížečům na celém světě většinu internetových stránek.
API	application programming interface, což znamená rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství v programování. API určuje, jakým způsobem se funkce knihovny mají volat ze zdrojového kódu programu
CCS	je zkratka pro anglický název Cascading Style Sheets, česky tabulky kaskádových stylů. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.
CGI	je protokol pro propojení externích aplikací s webovým serverem.
ColdFusion	je značkovací programovací jazyk pro aplikační vrstvu používaný hlavně pro webové aplikace.
C++	je objektově orientovaný programovací jazyk, který vyvinul Bjarne Stroustrup a další v Bellových laboratořích AT&T rozšířením jazyka C. C++ podporuje několik programovacích stylů (paradigmat) jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V současné době patří C++ mezi

	nejrozšířenější programovací jazyky.
DOM	je objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je API umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí.
DLL	je v programování funkční logický celek, který poskytuje služby pro programy. Většinou se jedná o sbírku procedur, funkcí a datových typů, či při objektově orientovaném přístupu o sadu tříd, uložených v jednom diskovém souboru.
FTP	File Transfer Protocol je protokol aplikační vrstvy z rodiny TCP/IP, je určen pro přenos souborů mezi počítači, na kterých mohou běžet velmi rozdílné operační systémy.
GUI	je druh komunikace s počítačem mající podobu interaktivních grafických prvků.
HTML	Hyper Text Markup Languege jazyk pro vytváření www stránek
HTTP	Hyper Text Transfer Protocol internetový protokol pro výměnu hypertextových dokumentů ve formátu HTML
HTTPS	je nadstavba počítačového protokolu HTTP, která poskytuje zvýšenou bezpečnost před odposloucháváním či podvržením dat.
Internet	je celosvětová počítačová „supersíť“, která spojuje jednotlivé menší sítě, pomocí sady protokolů IP.
Intranet	je počítačová síť, která používá stejné technologie (TCP/IP, HTTP) jako internet. Je ale „soukromá“. To znamená že je určena pro použití pouze malé skupiny uživatelů (například pracovníci nějakého podniku).
Java	Objektově orientovaný programovací jazyk
JDBC	Java Database Connectivity (známé spíše jako JDBC) je API pro programátory v programovacím jazyku Java, které definuje jednotné rozhraní pro přístup k relačním databázím.
JSP	byla vyvinutá společností Sun Microsystems, za účelem vývoje

	aplikací na straně servera. JSP soubory jsou vlastně HTML stránky, do kterých jsou vložené speciální tagy obsahující javovský zdrojový kód. Tento kód potom vytváří dynamický obsah stránky.
JVM	Java Virtial Machina virtuální stroj Javy
J2EE	je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů.
Kernel	se v počítačové terminologii nazývá jádro operačního systému, tedy program, který koordinuje činnost ostatních programů a zprostředkovává jim prostředky počítače. Název pochází z angličtiny, kde kernel znamená obecně „jádro“.
Mac OS	Macintosh Operating System – je označení operačního systému pro počítače Macintosh firmy Apple.
Mathematica	software pro počítání matematických funkcí.
MS Visual .NET Studio	je balík softwarových produktů, nástrojů a technologií pro rychlou a produktivní tvorbu aplikací využívajících prostředí produktové řady Microsoft Windows.
OS/2	byl operační systém firmy IBM.
PHP	je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek.
Server	je v informatice obecné označení pro počítač (tedy hardware) nebo proces, který poskytuje nějakou službu
SSI	jde o nejstarší a nejrozšířenější druh vkládaných vsuvek.
Unix	je víceúlohový a víceuživatelský operační systém, který je implementován na mnoha hardwareových platformách.
VBScript	je skriptovací jazyk Microsoft Visual Basic Scripting Edition určený pro vkládání kódu do webových stránek a běžné skriptování ve WSH (Interpreter skriptů nazývaný Windows

	Scripting Host), založený na jazyce Visual Basic.
Visual Basic	je dialektem programovacího jazyka BASIC od společnosti Microsoft.
webMathematica	je jednoduchý software jak vytvářet interaktivní matematické výpočty na webu.
Web server	Počítač, který je odpovědný za vyřizování požadavků HTTP od klientů - programů zvaných webový prohlížeč. Vyřízením požadavků se rozumí odeslání webové stránky. Webové stránky jsou obvykle dokumenty HTML.
Windows	je řada grafických víceúlohových operačních systémů společnosti Microsoft.
WWW	ve volném překladu „Celosvětová pavučina“, je označení pro aplikace internetového protokolu HTTP. Je tím myšlena soustava propojených hypertextových dokumentů.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÝ KÓD SOMA.JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>
<%@ page contentType="text/html; charset=windows-1250"%>
<html>
  <head>
    <title>Diferenciální evoluce v prostředí WebMathematica</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></head>
  <body background="../side_data/background.jpg">
    <h1>SOMA</h1>
 <style type="text/css">
  td    {font-size: 14px;}
  .parametry td {font-size: 16px;}
  .grafy td {font-size: 16px;}
</style>
 <form action="soma.jsp" method="post">
<msp:allocateKernel> Funckce pro optimalizaci
<select name="varFce" size="1">
<option value="17"> Vaše funkce (prosím vyplnit v dalším boxu)
<option value="1"> De Jong 1st
<option value="2"> De Jong 2nd
<option value="3"> De Jong 3rd
<option value="4"> De Jong 4th
<option value="5"> Rastrigin
<option value="6"> Schwefel
<option value="7"> Griewangk
<option value="8"> Sine Wave
<option value="9"> Stretched Sine
<option value="10"> Test Function Ackley
<option value="11"> Ackley
<option value="12"> Egg Holder
<option value="13"> Rana
<option value="14"> Pathological
<option value="15"> Michalewicz
```

<option value="16"> Masters

</select>

nebo

vaše vstupní funkce:

<input type="text" name="varYourFce" value="<msp:evaluate>MSPValue[$$varYourFce, "N[Sum[x[[i]]^2, {i, 1, Dimensions[x][[1]]}]]"]</msp:evaluate>" size="50" />

(Vstupní parametry jsou vektory {x1,x2,x3}.Prosím použijte syntaxi Mathematica x[[1]], x[[2]] etc.
 Počet parametru jsou vstupy Specimenu, stejně jako interval použití.)

Vstupní NP:

<input type="text" name="varNP" value="10" size="5" />

Vstupní Mass:

<input type="text" name="varMass" value="3" size="3" />

Vstupní PRT:

<input type="text" name="varPRT" value="0.1" size="3" />

Vstupní Step:

<input type="text" name="varStep" value="0.11" size="3" />

Vstupní Specimen:

<input type="text" name="varSpecimen" value="{{{R, {-1, 2}}}, {{Int, {-2, 3}}}} " size="30" />

Vstupní Migrace:

<input type="text" name="varMigrations" value="5" size="5" />

Vstupní AcceptedError:

<input type="text" name="varAcceptedError" value="-1" size="3" />

<msp:evaluate>

Needs["MSP`"];

Needs["MSP`HTML`"];

SetSecurity[];

<< AGO`Functions`

</msp:evaluate>

<msp:evaluate>

ToExpression["fce[x_]:="<>ToString[$$varYourFce]];

</msp:evaluate>

<msp:evaluate>

switchfunction[var_] := Switch[var, 1, DeJong1st, 2, DeJong2nd,3,DeJong3rd,4,DeJong4th,5,Rastrigin,6,Schwefel,7,Griewang,

8,SineWave,9,StretchedSine,10,TestFunctionAckley,11,Ackley,12,EggHolder,13,Rana,14, Pathological,15,Michalewicz,16,Masters,17,fce];

</msp:evaluate>

<msp:evaluate>

CostFunction= switchfunction[ToExpression[$$varFce]];

</msp:evaluate>

<msp:evaluate>

DoPopulation[PopSize_, 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, PopSize}];

  cv = CostFunction /@ pop;

  Return[MapThread[{#1, #2} &, {cv, pop}]]];

```
</msp:evaluate>
<msp:evaluate>
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[ind[[2]]];
   Print[str];
   Return[Flatten[{best, ind}, 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]]}]}]];


MinimalDiversity[pop_] := Module[{best, ind, str},
   best = pop[[Position[{##}, Min[##]][[1]][[1]] & @@ Transpose[pop][[1]]]][[1]];
   worst = pop[[Position[{##}, Max[##]][[1]][[1]] & @@ Transpose[pop][[1]]]][[1]];
   Return[If[Abs[best - worst] < MinDiv, {FinPop = {pop}; Abort[]}, pop]]];


SOMAATO[Pop_] := MinimalDiversity[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[Random[] < PRT, 1, 0], {i, Dim}])]], {t, 0,
PathLength, Step}] & /@ Pop), {2}])))]];
</msp:evaluate>
<msp:evaluate>
PopSize=If[ MSPValueQ[$$varPopSize],MSPToExpression[$$varPopSize]];
PathLength=If[ MSPValueQ[$$varPathLength],MSPToExpression[$$varPathLength]];
PRT=If[ MSPValueQ[$$varPRT],MSPToExpression[$$varPRT]];
```

```
Step=If[ MSPValueQ[$$varStep],MSPToExpression[$$varStep]];
Migrations=If[ MSPValueQ[$$varMigrations],MSPToExpression[$$varMigrations]];
MinDiv=If[ MSPValueQ[$$varMinDiv],MSPToExpression[$$varMinDiv]];
Specimen=If[ MSPValueQ[$$varSpecimen],MSPToExpression[$$varSpecimen]];
Specimen=Specimen /. {R -> Real, Int -> Integer};
Dim = Dimensions[Specimen][[1]];
</msp:evaluate>
<br>
První populace
<br>
<msp:evaluate>
Population=DoPopulation[PopSize,Specimen];
If[MSPValueQ[$$varMigrations],HTMLTableForm[Transpose[Population]]]
</msp:evaluate>
<br>
Finální populace
<br>
<msp:evaluate>
FinalPopulation = NestList[SOMAATO, Population,Migrations];
If[MSPValueQ[$$varPopSize], HTMLTableForm[Transpose[Last[FinalPopulation]]]]
</msp:evaluate>
<br>
Nejlepší jedinec
<br>
<msp:evaluate>
If[MSPValueQ[$$varPopSize]," na pozici " <>
ToString[BestInd[Last[FinalPopulation]][[1]]] <> " s hodnotou funkce " <>
ToString[BestInd[Last[FinalPopulation]][[2]]] <> " a s parametry " <>
ToString[BestInd[Last[FinalPopulation]][[3]]] <> "."]
</msp:evaluate>
<msp:evaluate>
If[MSPValueQ[$$varPopSize],BestInd[Last[FinalPopulation]]]
</msp:evaluate>
<msp:evaluate>
```

```
l1 = ListPlot[Table[Min[Transpose[FinalPopulation[[i]]][[1]]],
{i,Dimensions[FinalPopulation][[1]]}],PlotRange -> All, PlotJoined -> True, Axes ->
False, PlotLabel -> StyleForm["Output of Mathematica DE version",FontSize -> 14,
FontColor -> RGBColor[1, 0, 0]], TextStyle -> {FontWeight -> "Bold", FontColor ->
RGBColor[0, 0, 0.6], FontSize -> 12}, Frame -> True, RotateLabel -> False, FrameLabel -
> {"Number of Migrations", "Cost Value", "History of the Best Individual", ""}, ImageSize
-> 500];
```

</msp:evaluate>

<msp:evaluate>

```
l2 = ListPlot[Table[Max[Transpose[FinalPopulation[[i]]][[1]]],
{i,Dimensions[FinalPopulation][[1]]}], PlotRange -> All, PlotJoined -> True, Axes ->
False, PlotStyle -> RGBColor[1, 0, 0], RotateLabel -> False,PlotLabel ->
StyleForm["Output of Mathematica SOMA version", FontSize -> 14, FontColor ->
RGBColor[1, 0, 0]], TextStyle -> {FontWeight -> "Bold", FontColor -> RGBColor[0, 0,
0.6], FontSize -> 12}, Frame -> True, FrameLabel -> {"Number of Migrations", "Cost
Value", "History of the Worst Individual", ""}, ImageSize -> 500];
```

</msp:evaluate>

<msp:evaluate>

```
graphs = Show[l2, l1, FrameLabel -> {"Number of Migrations", "Cost Value", "History of
Both Individual", ""}];
```

</msp:evaluate>

graf

<msp:evaluate>

```
If[MSPValueQ[$$varPopSize], MSPShow[graphs]]
```

</msp:evaluate>

</msp:allocateKernel>


```
<input type="submit" name="submitButton"
                value="Spočítej" />
  </body>
</html>
```

PŘÍLOHA P II: ZDROJOVÝ KÓD DE.JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>
<%@ page contentType="text/html; charset=windows-1250"%>
<html>
<head>
   <title>Diferencialni evoluce v prostředi WebMathematica</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></head>
<body background="../side_data/background.jpg">
 <style type="text/css">
 td    {font-size: 14px;}
 .parametry td {font-size: 16px;}
 .grafy td {font-size: 16px;}
</style>
<h1>Diferenciání evoluce</h1>
<form action="de.jsp" method="post">
<msp:allocateKernel> Funckce pro optimalizaci
<select name="varFce" size="1">
<option value="17"> Vaše funkce (prosím vyplnit v dalším boxu)
<option value="1"> De Jong 1st
<option value="2"> De Jong 2nd
<option value="3"> De Jong 3rd
<option value="4"> De Jong 4th
<option value="5"> Rastrigin
<option value="6"> Schwefel
<option value="7"> Griewangk
<option value="8"> Sine Wave
<option value="9"> Stretched Sine
<option value="10"> Test Function Ackley
<option value="11"> Ackley
<option value="12"> Egg Holder
<option value="13"> Rana
<option value="14"> Pathological
<option value="15"> Michalewicz
```

<option value="16"> Masters

</select>

nebo

vaše vstupní funkce:

<input type="text" name="varYourFce" value="<msp:evaluate>MSPValue[$$varYourFce, "N[Sum[x[[i]]^2, {i, 1, Dimensions[x][[1]]}]]"]</msp:evaluate>" size="50" />

(Vstupní parametry jsou vektory {x1,x2,x3}.Prosím použijte syntaxi Mathematica x[[1]], x[[2]] etc.
 Počet parametru jsou vstupy Specimenu, stejně jako interval použití.)

Vstupní NP:

<input type="text" name="varNP" value="10" size="5" />

Vstupní F:

<input type="text" name="varF" value="1" size="3" />

Vstupní CR:

<input type="text" name="varCR" value="0.8" size="3" />

Vstupní Specimen:

<input type="text" name="varSpecimen" value="{{{R, {-1, 2}}}, {{Int, {-2, 3}}}} " size="30" />

Vstupní generace:

<input type="text" name="varGenerations" value="5" size="5" />


```
<msp:evaluate>
Needs[ "MSP`"];
Needs["MSP`HTML`"];
SetSecurity[];
<< AGO`Functions`
</msp:evaluate>
<msp:evaluate>
ToExpression["fce[x_]:="<>ToString[$$varYourFce]];
</msp:evaluate>
<msp:evaluate>
switchfunction[var_] := Switch[var, 1, DeJong1st, 2,
DeJong2nd,3,DeJong3rd,4,DeJong4th,5,Rastrigin,6,Schwefel,7,Griewang,
8,SineWave,9,StretchedSine,10,TestFunctionAckley,11,Ackley,12,EggHolder,13,Rana,14,
Pathological,15,Michalewicz,16,Masters,17,fce];
</msp:evaluate>
<msp:evaluate>
CostFunction= switchfunction[ToExpression[$$varFce]];
</msp:evaluate>
<msp:evaluate>
DoPopulation[NP_, 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,NP}];
   cv = CostFunction /@ pop;
   Return[MapThread[{#1, #2} &, {cv, pop}]]]
</msp:evaluate>
<msp:evaluate>
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[ind[[2]]];
   Print[str];
   Return[Flatten[{best, ind}, 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]]}]}]];

SelectOtherRand1Bin[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,
P}], {i, 3}]];

   Return[rand];

DERand1Bin[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[SelectOtherRand1Bin[#2[[1]]] &, Pop])}, 1])))}];

</msp:evaluate>

<msp:evaluate>

NP=If[ MSPValueQ[$$varNP],MSPToExpression[$$varNP]];

F=If[ MSPValueQ[$$varF],MSPToExpression[$$varF]];

Cr=If[ MSPValueQ[$$varCR],MSPToExpression[$$varCR]];

Generations=If[ MSPValueQ[$$varGenerations],MSPToExpression[$$varGenerations]];

Specimen=If[ MSPValueQ[$$varSpecimen],MSPToExpression[$$varSpecimen]];

Specimen=Specimen /. {R -> Real, Int -> Integer};

Dim = Dimensions[Specimen][[1]];

</msp:evaluate>

<br>

První populace

<br>

<msp:evaluate>

Population=DoPopulation[NP,Specimen];

If[MSPValueQ[$$varGenerations],HTMLTableForm[Transpose[Population]]]

</msp:evaluate>

<br>

Finální populace

<br>

<msp:evaluate>
```

```
FinalPopulation = NestList[DERand1Bin, Population,Generations];

If[MSPValueQ[$$varNP], HTMLTableForm[Transpose[Last[FinalPopulation]]]]

</msp:evaluate>

<br>

Nejlepší jedinec

<br>

<msp:evaluate> If[MSPValueQ[$$varNP]," na pozici "

<> ToString[BestInd[Last[FinalPopulation]][[1]]] <> " s hodnotou funkce " <>
ToString[BestInd[Last[FinalPopulation]][[2]]] <> " a parametry " <>
ToString[BestInd[Last[FinalPopulation]][[3]]] <> "."] </msp:evaluate> <msp:evaluate>

If[MSPValueQ[$$varNP],BestInd[Last[FinalPopulation]]] </msp:evaluate>
<msp:evaluate>

l1 = ListPlot[Table[Min[Transpose[FinalPopulation[[i]]][[1]]], {i,
Dimensions[FinalPopulation][[1]]}],PlotRange

-> All, PlotJoined -> True, Axes -> False, PlotLabel -> StyleForm["Output of Mathematica

DE version",FontSize -> 14, FontColor -> RGBColor[1, 0, 0]], TextStyle -> {FontWeight

-> "Bold", FontColor -> RGBColor[0, 0, 0.6], FontSize -> 12}, Frame -> True,
RotateLabel -> False, FrameLabel -> {"Number of Generations", "Cost Value", "History of
the Best Individual", ""}, ImageSize -> 500]; </msp:evaluate> <msp:evaluate> l2 =

ListPlot[Table[Max[Transpose[FinalPopulation[[i]]][[1]]],
{i,Dimensions[FinalPopulation][[1]]}], PlotRange -> All, PlotJoined -> True, Axes ->
False, PlotStyle -> RGBColor[1, 0, 0], RotateLabel -> False,PlotLabel ->
StyleForm["Output of Mathematica DE version", FontSize -> 14, FontColor ->
RGBColor[1, 0, 0]], TextStyle -> {FontWeight -> "Bold", FontColor -> RGBColor[0, 0,
0.6], FontSize -> 12}, Frame -> True, FrameLabel -> {"Number of Generations", "Cost
Value", "History of the Worst Individual", ""}, ImageSize -> 500]; </msp:evaluate>
<msp:evaluate> zkouska = Show[l2, l1, FrameLabel -> {"Number of Generace", "Cost
Value", "History of Both Individual", ""}]; </msp:evaluate>

<br>graf<br>

<msp:evaluate>

If[MSPValueQ[$$varNP],

    MSPShow[zkouska]]

</msp:evaluate>

</msp:allocateKernel>

<br>

<input type="submit" name="submitButton"

            value="Spočítej" />

 </body></html>
```

PŘÍLOHA P III: ZDROJOVÝ KÓD INDEX.HTM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE">
<meta http-equiv="Content-Language" content="cs">
<title>webMATHEMATICA - Evoluční algoritmy</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
</head>
<body background="en/side_data/pozadi.jpg" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%"  border="0" height="100%">
<tr>
<td height="76" colspan="2" width="100%">
<div align="center">
<p>
<img border="0" src="cz/side_data/ea.jpg" width="400" height="60"><img border="0" src="cz/side_data/webm-white-animated.gif" width="400" height="60"></p>
<p align="left">
<a href="index.htm"><img src="cz/side_data/cz.gif" width="36" height="24" border="0"></a>
<a href="index_eng.htm"><img src="cz/side_data/eng.gif" width="49" height="24" border="0"></a></p>
</div></td>
</tr>
<tr>
<td width="250" height="100%">
<iframe src="cz/treemenu_data/treemenu.html" width="248" height="100%" frameborder="0"  name="I1"></iframe>
</td>
<td width="100%" rowspan="2" height="100%">
<iframe src="cz/uvod.htm" name="main" frameborder="0" height="100%" width="100%"></iframe>
</td></tr>
</table></body></html>
```