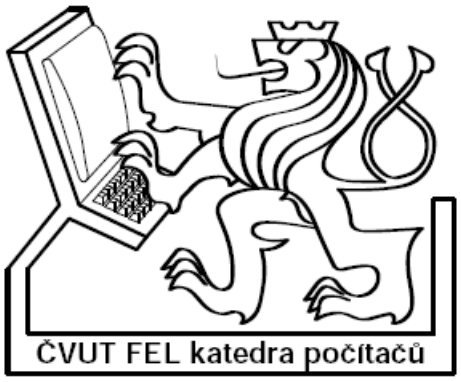
České vysoké učení technické v Praze
Fakulta elektrotechnická



Bakalářská práce

Vizualizace běhu genetických algoritmů

Petr Benhák

Vedoucí práce: Ing. Petr Fišer

Studijní program: Elektrotechnika a informatika strukturovaný bakalářský

Obor: Informatika a výpočetní technika

Červen 2006

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 26.6.2006 .....

Anotace

Práce popisuje základní rysy genetických algoritmů a jejich aplikaci pro řešní NP-těžkých problémů. Podává výklad jejich principů a postupů při jejich programování. Pro demonstraci běhu genetického algoritmu je součástí práce také program ve formě Java apletu, který umožňuje experimentování s nastavením různých parametrů genetického algoritmu. Program řeší několik problémů, jsou to: problém batohu, problém obchodního cestujícího, problém splnitelnosti booleovské formule a problém maximalizace funkce. Všechny tyto problémy jsou v textu práce podrobněji popsány. Součástí jsou také testy, které mají za cíl porovnat různá nastavení parametrů genetického algoritmu. Tyto testy byly provedeny s pomocí demonstračního programu a jejich výsledky jsou v textu shrnuty.

Abstract

This work describes basic features of genetic algorithms and their application in NP-hard problems solving. The work gives explanation of their principles and programming processes. In order to show genetic algorithm run, the work includes Java applet to allow users modify and experiment on algorithm settings. Program solves several problems, these are knapsack problem, travelling salesman problem, boolean satisfiability problem and function maximization problem. All these problems are described in more detail in the text. The work also includes tests, which aims to compare different settings of parameters of genetic algorithm. These tests were performed with help of demonstrational program and the results summary is in the further text.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem práce je podat výklad problematiky genetických algoritmů. K tomuto účelu jsem vytvořil Java aplet, který demonstruje běh jednoduchého genetického algoritmu. Uživateli je umožněno měnit různé parametry genetického algoritmu a pozorovat změny v jeho chování. Pro snažší vyhodnocení chování je vývoj populace vyobrazen ve formě grafu, který zahrnuje nejdůležitější ukazatele, jako je maximální, minimální a průměrný fitness v jednotlivých generacích.

Přestože princip evoluce je aplikovatelný na široké spektrum problémů, v jednotlivostech se pro různé problémy liší. Principy evoluce můžeme dokonce použít, pokud nevíme, jak problém exaktně řešit nebo pokud ho exaktně řešit nechceme.

Pro demonstraci genetického algorimu jsem s ohledem na účel práce, která by měla být úvodem do genetických algoritmů, vybral několik problémů a implementoval je do testovacího programu. Je to problém batohu, problém splnitelnosti booleovské formule, problém obchodního cestujícího a hledání maxima funkce na zadaném intervalu.

Funkce implementovaného genetického algoritmu je podložena testy, které kombinují různé parametry algoritmu. Testy byly navrženy tak, aby ukázaly, jak volit parametry genetického algoritmu pro dosažení dobrých výsledků.

2 Struktura práce

Kapitola 3 je věnována rozdělení výpočetních problémů. V kapitole 4 je rozebrán průběh jednoduchého genetického algoritmu, je zde uveden pseudokód, jehož struktury se drží i demonstrační program. V kapitole 6 je popsán princip jednotlivých metod selekce. V této kapitole je také popsáno pomocí pseudokódu, jak tyto metody implementovat. Metody křížení a mutace jsou popsány v kapitole 7 a 8. Poté následují různé metody používané při programování genetického algoritmu, jako je elitářství a škálování. Dále v textu (kapitola 12) najdete základní větu genetických algoritmů, základy na kterých stojí a její důsledky. V kapitole 15 naleznete popis některých metod lokálního prohledávání. Kapitola 16 je již věnována popisu implementace testovacího programu a problémům, které řeší. U každého z těchto problémů je popsán princip řešení pomocí genetického algoritmu a shrnutí výsledků provedených testů. Vlastní testy se nachází v dodatku A.

3 Rozdělení problémů

Výpočetní problémy můžeme rozdělit do následujících skupin:

- Rozhodovací problémy *(decision problems)* - jsou to problémy, na které můžeme odpodědět ano nebo ne.
  Příklady rozhodovacího problému: *Existuje Hamiltonovská kružnice v daném grafu ? Existuje takové přiřazení logických hodnot, aby byla formule pravdivá ?*

- Optimalizační problémy *(optimization problems)* - velké množství praktických problémů nejsou rozhodovací problémy, ale právě optimalizační. Tento druh problémů řeší genetický algoritmus. U optimalizačních problémů se snažíme nalézt nejlepší řešení podle nějakého kritéria. Cílem optimalizačního problému je nalézt takové přípustné řešení, které má minimální (nebo maximální) ohodnocení. Abychom mohli využívat teorie NP-úplnosti pro tuto třídu problémů, musíme zadání upravit tak, aby nový problém byl rozhodovací. To můžeme udělat přidáním omezení K k instanci problému. Rozhodovací problém můžeme pak formulovat takto: *Existuje řešení problému s ohodnocením K ?*

  Příklad: *Najdi takové pravdivostní ohodnocení, aby co nejvíce klauzulí ve formuli bylo splněno.*

- Prohledávací problémy *(search problems)* - úkolem je nalézt řešení splňující jisté vlastnosti.

  Příklad: *Najdi takové pravdivostní ohodnocení pro danou formuli, aby byla tato formule pravdivá.*

- Problémy určující počet řešení *(counting problems)* - úkolem je určit počet možných řešení.

3.1 Třídy složitosti

Mnohé problémy jsou výpočetně jednoduché, to znamená, že existuje polynomiální algoritmus, který tento problém řeší. Délka výpočtu tímto algoritmem se zvyšuje polynomiálně v závislosti na velikosti vstupu (instance). Naproti tomu existují problémy, které jsou výpočetně složité, neexistuje pro ně algoritmus, řešící je v polynomiálním čase. Délka výpočtu tohoto algoritmu se zvyšuje rychleji, než polynomiálně.

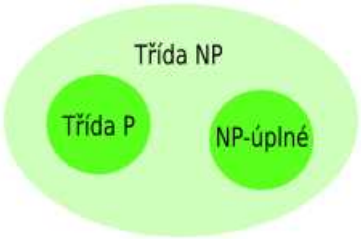
Řekneme, že problém B může být převeden (redukován) na problém A ($B \sqsubseteq A$), pokud existuje tranformace T, která pro každou vstupní instanci V algoritmu B vytvoří vstupní instanci T(V) algoritmu A tak, aby platilo, že pro každý vstup T(V) poskytuje algoritmus A ekvivalentní výstup jako algoritmus B pro vstup V. Aby bylo redukce přínosná, je třeba ji provést v polynomiálním čase.

Všechny problémy můžeme rozdělit do následujících tříd:

- Rozhodovací problémy deterministicky řešitelné v polynomiálně omezeném čase, se označují jako třída P *(Polynomial)*.
- Třída NP *(Nondeterministic Polynomial)*, takové rozhodovací problémy, jejichž správné řešení může být ověřeno v polynomiálním čase nebo ekvivalentně jejichž řešení může být nalezeno nedeterministicky v polynomiálním čase.
- Problémy řešitelné s vyšší než polynomiální složitostí.

Otázka, která není dosud vyřešena je, zda P=NP. Je zřejmé, že všechny P problémy jsou také NP, ale obráceně to neplatí. P je tedy podmnožina NP.

Množinu problémů L takových, že každý NP problém může být redukován na L v polynomiálním čase, nazýváme **NP-těžké** problémy. Jsou tedy nejméně tak těžké, jako jakýkoliv problém v NP. Pokud jsou navíc v NP nazýváme je **NP-úplnými** problémy. Situaci znázorňuje následující diagram:



Obrázek 1. Souvislost mezi P, NP, NP-úplnými problémy

Pokud existuje efektivní algoritmus k řešení jednoho NP-úplného problému, potom všechny NP-úplné problémy jsou řešitelné efektivním algoritmem (v polynomiálním čase).

Základním problémem ve třídě NP je problém splnitelnosti booleovské formule, který odpovídá na otázku: *Existuje pro danou booleovskou formuli o n proměnných takové přiřazení hodnot jednotlivým proměnným, aby celá formule byla pravdivá ?* Tento problém je NP-úplným, proto můžeme libovolný problém z NP převést na ekvivalentní SAT problém v polynomiálním čase. Opačná transformace v polynomiálním čase nemusí existovat.

4 Genetické algoritmy obecně

Genetické algoritmy představují výpočetní model inspirovaný evoluční teorií. Základní myšlenku evoluce můžeme vyjádřit asi takto: *"Jedinci jednoho druhu se v populaci trochu odlišují, jejich znaky se přenášejí na další generace. Všechny druhy mají nadbytek potomstva. V boji o život přežívají pouze ti nejschopnější, méně schopní hynou".*

Obdobně pracují i genetické algoritmy, simulují evoluci. Nadprůměrné vlastnosti jedinců přežívají a navzájem se kombinují, špatné vlastnosti naopak přirozeným výběrem postupně z populace mizí.

Takto lze řešit složité problémy i bez znalosti přesného způsobu exaktního řešení. Slovo řešit ve spojitostí s genetickým algoritmem znamená spíše hledat "dobré řešení", protože zdaleka ne vždy je nalezené řešení skutečně to optimální. Přesnost řešení můžeme do jisté míry ovlivnit nastavením vhodných parametrů genetického algoritmu pro daný problém, přesto hraje významnou roli mnoho náhodných faktorů, především rozložení jedinců v počáteční populaci, to jak proběhne křížení a jaké štěstí budeme mít při mutaci.

Genetický algoritmus je **globální optimalizační metoda**, založená na populaci jedinců konstantní velikosti. Každý jedinec je reprezentován konečným řetězcem symbolů označovaným jako chromozóm, který kóduje potenciální řešení v prohledávaném prostoru všech možných řešení.

O použití genetického algoritmu uvažujeme, pokud je prohledávaný prostor příliš rozsáhlý nebo o neznámé velikosti. V takovém případě nemusí být reálné kompletní prohledání celého prostoru. Přestože mnoho problémů lze řešit např. pomocí dynamického programování za cenu nárůstu paměťových nároků, je genetický algoritmus dobrou volbou i pro tyto problémy, pokud nepotřebujeme získat opravdu to nejlepší řešení, ale spokojíme se s řešením, které se k nejlepšímu jen blíží.

V tomto textu dále uvedu několik příkladů, kde použití genetických algoritmů umožní řešení problémů, které jsou jen těžko řešitelné pomocí konvenčních metod.

4.1 Průběh genetického algoritmu

Na počátku výpočtu vytvoříme typicky populaci náhodných jedinců reprezentujících náhodné vzorky v prohledávaném prostoru. Lze také vytvořit populaci podle nějaké heuristiky.

Každý krok genetického algoritmu vede k vytvoření nové sady jedinců. Z jedné populace vytvoříme novou (přejdeme do další generace) tak, že dekódujeme jedince z dané populace a podle nějaké předem definované funkce jim přiřadíme hodnotu, která bude reprezentovat kvalitu jedince podle námi definovaných parametrů. Výslednou hodnotu můžeme upravit s ohledem na ostatní jedince v populaci a nazveme ji **fitness**. Ohodnocení by mělo být velmi rychlé, protože se provádí na mnoha jedincích v mnoha generacích. Výsledná hodnota fitness vyjadřuje kvalitu řetězce a je zásadní pro výběr jedinců do další generace. Do příští generace se mají větší šanci dostat jedinci s větší hodnotou fitness. Po výběru jedinců (selekce) provedeme za určitých podmínek vzájemné křížení a mutaci, nakonec vytvoříme novou populaci.

Vytvoření nové populace je možné rozložit do dvou kroků. Nejprve vytvoříme *"mezipopulaci"* pomocí selekce. Konečnou populaci vytvoříme aplikací křížení a následně mutace na "mezipopulaci". Křížit budeme vždy dva náhodně vybrané jedince z mezipopulace s určenou pravděpodobností křížení. Tato abstrakce je užitečná pro "matematický popis" genetického algoritmu. Proces přechodu od současné populace k nové tvoří jednu generaci běhu genetického algoritmu.

Algoritmus probíhá podle následujícího schématu:

1. Vytvoř náhodnou populaci o J jedincích. Každý jedinec v populaci je tvořen řetězcem o n bitech (inicializace).
2. Přiřaď fitness každému jedinci v populaci. Fitness vyjadřuje šance jedince k přežití. Čím je jedinec kvalitnější, tím větší hodnotu fitness bude mít.
3. Dokud není vytvořeno J nebo J-1 potomků, opakuj následující:
   I. Vyber dvojici jedinců (rodiče) ze současné populace. Pravděpodobnost výběru je rostoucí funkcí fitnessu. Jeden jedinec může být rodičem více než jednou.
   II. Proveď křížení rodičů s danou pravděpodobností. Výsledkem křížení jsou dva noví potomci. V případě, že ke křížení nedojde, vytvoř pouze kopie rodičů a považuj je za potomky.
   III. Proveď na nově vzniklých jedincích mutaci s danou pravděpodobností.
   IV. Nově vzniklé jedince umísti do nové populace.
4. Pokud je velikost nové populace J-1 (původní populace obsahuje liché množství jedinců), lze přidat náhodného jedince k vytvoření kompletní populace.
5. Nahraď starou populaci populací novou a jdi do bodu 2.

Popsaný průběh genetického algoritmu (také např. v [MIT]) je označován jako **jednoduchý genetický algoritmus** *(Simple Genetic Algorithm SGA, Goldberg 1989)*. Jednoduchý genetický algoritmus se vyznačuje typickými vlastnostmi podle *tabulky 1*.

vlastnost	realizace
reprezentace jedinců	binární řetězce
inicializace populace	vytvoření náhodných řetězců
křížení	křížení v jednom bodě s určenou pravděpodobností
mutace	změna bitu v řetězci s určenou pravděpodobností
výběr rodičů	pravděpodobnost výběru je úměrná fitness
výběr přeživších	všichni potomci nahradí rodiče

Tabulka 1. Vlastnosti SGA.

Pokud usoudíme, že nalezené řešení je již dostatečně dobré, ukončíme běh algoritmu v bodě 2. Ukončovací podmínka může být definována různě. Intuitivně platí, že dobrá ukončovací podmínka zastaví běh algoritmu v případě, kdy již neočekáváme rozumné vylepšení, popřípadě pokud by náklady na další výpočet převýšily náklady, které jsme ochotni obětovat pro dosažení očekávaného řešení. Různé varianty pro ukončení mohou být:

- Po uplynutí stanoveného času nebo počtu generací.
- Při dosažení předem stanovené kvality řešení.
- Pokud nebylo dosaženo žádného zlepšení po definovaný počet generací.
- Pokud se zdá, že algoritmus konvergoval. Zda došlo ke konvergenci můžeme odpozorovat například podle počtu shodných jedinců v populaci.

Jednotlivé kroky algoritmu si podrobněji rozebereme v následujícím textu. Samozřejmě existuje mnoho modifikací, které se svou implementací mohou lišit ve všech popsaných bodech. Ve skutečnosti nazýváme každý výpočetní model založený na populaci jedinců, který používá operátorů křížení a mutace k prohledávání prostoru možných řešení, genetickým algoritmem. Příkladem jsou hybridní genetické algoritmy, které kombinují prostý genetický algoritmus s dalšími metodami, např. hill-climbing (viz. další text). Tyto algoritmy, závislé na řešeném problému, vesměs vykazují velmi dobré výsledky. Jiný pojem, který se objevuje ve spojitosti s genetickými algoritmy, je pojem evoluční algoritmus. Mezi evoluční algoritmy patří genetické algoritmy, evoluční programování, evoluční strategie a genetické programování.

4.2 Terminologie

Vzhledem k tomu, že genetické algoritmy vycházejí z evoluce, promítá se biologie i do používané terminologie ve spojitosti s genetickými algoritmy.

- Chromozóm - Živé organismy se skládají z buněk, každá buňka obsahuje chromozómy, které slouží jako stavební plán organismu. Chromozóm se skládá z genů. V genetických algoritmech rozumíme pod pojmem chromozóm reprezentaci možného řešení. Chromozómy bývají kódovány např. jako binární řetězce.
- Gen - Na geny můžeme pohlížet jako na kódování určité vlastnosti, například barvy očí. Různé vlastnosti, které gen určuje, nazýváme alely. V genetických algoritmech nazýváme genem jednu konfigurační proměnnou. Může to být jeden bit nebo i více přilehlých bitů kodující určitou část řešení. Alela v binárním řetězci může být 1 nebo 0.

- Lokus - Pozice na které se nachází daný gen v chromozómu.
- Genom - Kompletní genetický materiál, tedy všechny chromozómy nazýváme genom. Jedinci s identickými genomy mají stejný genotyp. Fenotyp je vnější projev, např. konkrétní barva očí.
- Jedinec - Odpovídá možnému řešení.

4.3 Inicializace

Na počátku běhu algorimu je nutné zvolit počáteční populaci. Nejběžnější bývá vytvoření náhodné populace. Tento přístup není ale vhodný pro všechny problémy. Může se stát, že náhodný řetězec kóduje nepřípustné řešení *(viz. problém batohu dále v textu)*. Pokud k takové situaci dojde, je nutné řetězec upravit tak, aby byl přípustný nebo ho nějakým způsobem penalizovat a snížit tak jeho šanci k přechodu do další generace. Stejný postup lze použít pro nepřípustný řetězec, který vznikl křížením či mutací. Abychom se při inicializaci takovýmto problémům vyhli, můžeme do počáteční populace umístit pouze jedince přípustné v závislosti na řešeném problému.

Kromě kódování pomocí binárních řetězců se používají i jiné druhy kódování, např. celočíselné, pomocí stromu, maticí. V těchto případech musíme vytvořit vhodné operátory křížení a mutace přizpůsobené zvolené datové struktuře.

V každém případě by počáteční populace měla být různorodá a pokrývat rovnoměrně celý stavový (prohledávaný) prostor. V opačném případě je radikálně snížena pravděpodobnost nalezení optimálního řešení.

5 Kódování a hodnotící funkce

Kódování je závislé na řešeném problému. Při použití binárního kódování si optimalizační problém mužeme představit jako "černou skříňku" s mnoha páčkami. Každá páčka může být ve dvou polohách a odpovídá jedné logické hodnotě v binárním řetězci. Informace, kterou nám skříňka může poskytnout, je pouze, jak kvalitní je dané nastavení. Tento výstup určuje ohodnocení kombinace vstupních proměnných. Hodnotící funkce je komponenta genetického algoritmu, která je závislá na řešeném problému. Naším cílem je nastavit takovou vstupní kombinaci proměnných, která získá nejvyšší ohodnocení.

Problémy, které řeší genetické algoritmy jsou obvykle nelineární. Z toho plyne, že vstupní proměnné jsou na sobě závislé a nemůžeme s nimi zacházet odděleně. Například izolovat jednu proměnnou a provést optimalizaci, poté optimalizovat druhou atd. S kombinací vstupních proměnných musíme zacházet jako s celkem, to znamená brát ohled na provázanost těchto proměnných.

V jednoduchém genetickém algoritmu se snažímě maximalizovat funkci c definovanou takto:

$$c\colon \{0,1\}^N \longrightarrow \mathbb{R}$$

Každý problém, který chceme řešit pomocí jednoduchého genetickího algoritmu, musí být zakódován binárně. Za předpokladu ohodnocení řešení pomocí funkce c definujeme v kanonickém genetickém algoritmu fitness funkci takto:

$f(s) = \frac{c(s)}{\bar{c}}$, kde $\bar{c}$ je průměr ze všech c(s).

Slovo fitness budu dále v textu používat v jeho širším významu a to kvality jedince, která odráží šance k reprodukci.

6 Selekce

Selekcí nazýváme výběr jedinců do další generace. Rozhoduje o tom, kdo přežije a dostane možnost k reprodukci. Existuje mnoho způsobů, jak selekci implementovat, některé způsoby jsou dále popsány (vážená ruleta, rank based, tournament, SUS).

V jednoduchém genetickém algoritmu nahrazujeme celou starou populaci populací novou. To ale nemusí být pravidlem. Jiné varianty algoritmu nemusí nahrazovat celou populaci. Selekci můžeme také vidět v kontextu výběru mezi rodiči a potomky pro vytvoření nové populace, tzv. *selekce přeživších*. Tuto můžeme rozdělit na selekci na základě stáří (možné implementovat například pomocí fronty) nebo na základě fitnessu (použijeme některou z dále popsaných selekčních metod).

6.1 Vážená ruleta

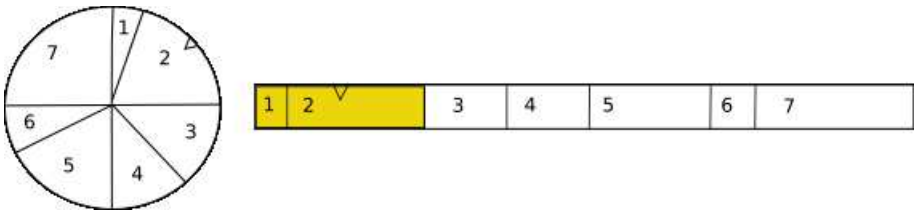
Asi nejpřirozenějším způsobem výběru je použití tzv. vážené rulety. Každý jedinec je reprezentován jednou výsečí rulety. Velikost výseče proporcionálně odpovídá honotě fitness jedince. Fitness přímo udává pravděpodobnost P vylosování jedince ruletou, $P = \frac{f_i}{\sum f_i}$. To zaručuje, že i slabší jedinci mají možnost uplatnit se v další generaci a zúčastnit se tak křížení a mutace. Může se stát, že malá změna méně kvalitního jedince či křížení s jiným jedincem poskytnou velice kvalitní řešení.

Tím, že ruletu roztočíme N-krát, získáme N řetězců do nové populace. Očekávaný počet jedinců I s pravděpodobností vylosování P v příští generaci určíme $I = P \cdot N$.

Tato metoda má také velkou nevýhodu. Pokud existuje v populaci jedinec s velkou hodnotou fitness oproti zbytku populace, může tento jedinec velmi rychle pokrýt celou populaci. Této nežádoucí situaci říkáme předčasná konvergence *(premature convergence)*. Typicky se takové kvalitnější řetězce vyskytují již na počátku běhu algoritmu, což značně redukuje možnost algoritmu nalézt dobré řešení. Dá se říct, že je kladen příliš velký důraz na prohledávání prostoru reprezentovaného vysoko ohodnocenými jedinci na úkor ostatních jedinců v populaci.

Abychom předešli předčasné konvergenci, můžeme použít tzv. škálování (scaling). Škálování je dále v textu popsáno.

Na obrázku je znázorněna ruleta s očíslovanými výsečemi. Největší výseč zabírá jedinec s číslem 7 a má největší šanci na vylosování. Vylosován byl ovšem jedinec s číslem 2.



Obrázek 2. Vážená ruleta

Implementaci selekce váženou ruletou si ukážeme na následujícím pseudokódu:

1. Spočti součet S hodnot fitness všech jedinců v populaci.
2. Vyber náhodné číslo R v intervalu <0, S), to graficky odpovídá umístění ukazatele rulety.
3. Procházej postupně jedince v populaci a sčítej jejich hodnoty fitness, dokud nebude součet větší než R.
4. Jedinec, jehož fitness po přičtení způsobí překročení hodnoty R, je vylosovaným jedincem.

6.2 Souboj (tournament)

Souboj mezi několika náhodně vylosovanými jedinci probíhá na základě hodnoty fitness. Z $K \geq 2$ náhodně vybraných jedinců vybereme toho, který má nejvyšší fitness (K=1 odpovídá náhodnému výběru). Zvyšováním hodnoty K dosáhneme většího selekčního tlaku a tím také rychlejší konvergence.

Tento způsob selekce se chová podobně jako selekce na základě ranku, ve smyslu selekčního tlaku, avšak je snadný na implementaci, rychlý a vhodný pro paralelizaci. V provedených testech se ukazuje jako nejlepší volba.

Varianta turnaje nazývaná pravděpodobnostní souboj (probabilistic tournament) vybere nejlepšího jedince s pravděpodobností p, druhého nejlepšího s pravděpodobností $p \cdot (1 - p)$, třetího nejlepšího s pravděpodobností $p \cdot (1 - p)^2$ atd. Pokud se p=1, jedná se o původní variantu vybrání nejlepšího jedince.

6.3 Stochastic universal sampling

Jedná se o jednofázovou metodu podobnou ruletě. Na rozdíl od rulety, kde v jednom průběhu vybereme jednoho jedince, zde jedním průběhem vybereme všechny jedince do další generace. Místo jednoho ukazatele na klasické ruletě použijeme N stejně vzdálených ukazatelů. Tím jedním otočením rulety vybereme celou novou populaci. Označíme-li fitness i-tého jedince v populaci jako f(i), můžeme průběh vyjádřit následujícím pseudokódem:

```
R = rand(0, SUM_OF_FITNESS/N) //umístění prvního ukazatele
SUM = 0
for I=1 to N do
   SUM = SUM + f(i)
   while(SUM > R) //dokud jsou ukazatele v téže výseči, provádíme výběr
       select(i)  //vyber jedince
       R = R + (SUM_OF_FITNESS/N)   //proveď posun ukazatele
```

SUS ovšem neřeší hlavní nedostatek vážené rulety a sice problém předčasné konvergence.

6.4 Rank ruleta

Problém předčasné konvergence se snaží řešit *"rank ruleta"*. Průběh můžeme rozdělit do dvou fází. Nejprve přiřadíme každému jedinci jisté ohodnocení nazývané rank. Po té použijeme vážené rulety, která vybere jedince na základě přiřazeného ranku.

Přiřazení ranku si lze představit následovně: Seřadíme jedince v populaci podle jejich hodnoty fitness. Jedinci s nejnižší hodnotou přiřadíme rank 1 a následujícímu o 1 větší. Poslední jedinec bude mít rank odpovídající velikosti populace. Pokud se vyskytuje stejný jedinec vícekrát, vždy bude mít stejný rank.

Nevýhodou této metody může být pomalejší konvergence díky tomu, že fitnessy kvalitních řetězců jsou si velmi podobné. Pokud jsou ale hodnoty původního fitness vysoké a podobné, což je běžná situace, může naopak přispět k rychlejší konvergenci. Průběh je tedy následující:

1. `Přiřaď každému jedinci rank na základě fitness.`
2. `Vyber jedince pomocí vážené rulety.`

7 Křížení

Účelem křížení řetězců je testovat nové části prohledávané oblasti namísto stálého testování jednoho bodu. Při křížení si jedinci vyměňují informace s cílem nalézt lepší řešení.

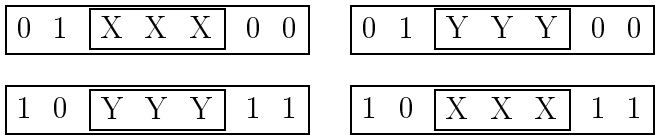
Poznámka 1. Křížení ovšem komplikuje efekt implicitního paralelismu. Může se stát, že potomek bude přesunut do jiné oblasti (opustí stavební blok) a zpomalí tak konvergenci (viz. kapitola 12).

Základní variantou křížení je tzv. **křížení v jednom bodě** *(one-point crossover)*. Křížení probíhá mezi dvěma jedinci (rodiči) a výsledkem jsou opět dva jedinci, potomci. Řekneme-li, že křížíme řetězce na 2. místě, znamená to křížení mezi druhým a třetím bitem řetězce. Zvolíme-li pozici křížení v řetězci, vytvoříme potomky tak, že bity před pozicí křížení v potomkovi pocházejí z jednoho rodiče a bity za touto pozicí z druhého. Nevýhodou je velká závislost na pozici genu v řetězci. Např. geny na opačných stranách řetězce budou vždy odděleny, sousední geny budou rozděleny jen s malou pravděpodobností (1/délka_řetězce). Tento problém nazýváme poziční závislost (positional bias). Metodou křížení v jednom bodě získáme dva potomky následujícím způsobem.

Rodiče:	Potomci:
X X X X 0 0 1 0	Y Y Y Y 0 0 1 0
Y Y Y Y 1 0 1 1	X X X X 1 0 1 1

Kkřížení ve dvou bodech *(two-point crossover)* probíhá následovně: Náhodně zvolíme dva body ve kterých se bude křížit a bity mezi nimi přehodíme. Křížení v jednom bodě je speciální případ křížení ve dvou bodech, kdy je vždy za počáteční bod volen první bit řetězců.

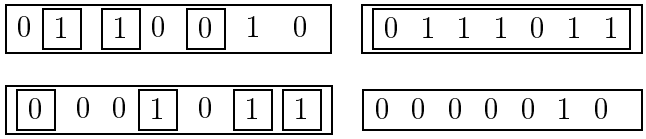
Rodiče: Potomci:



Možné je i použití křížení ve více bodech, ovšem díky velkému destruktivnímu charakteru, se toto křížení příliš nepoužívá.

Dalším způsobem křížení je tzv. **uniformní křížení** *(uniform crossover)*. Při tomto křížení procházíme každý bit rodičů a náhodně je mezi sebou prohazujeme. Při užití parametrizovaného uniformního křížení můžeme volit pravděpodobnost přehození bitů. Ta se nejčastěji volí v intervalu <0.5, 0.8>. Hodnota 0.5 odpovídá základní variantě náhodného přehození.

Rodiče: Potomci:



V malých populacích může uniformní křížení dosahovat lepších výsledků než klasické operátory křížení, protože pomáhá překonat nedostatečné pokrytí prohledávaného prostoru malou populací. V takovém případě bývají obecně více destruktivní operátory křížení úspěšnější.

8 Mutace

Mutace náhodně mění jedince v populaci. Na počátku běhu algoritmu specifikujeme pravděpodobnost, se kterou bude docházet k mutaci. S touto pravděpodobností bude docházet ke změně bitů jedinců v populaci. Procházen je každý bit každého řetězce a v případě, že v daném bitu dojde k mutaci, je tento znegován.

Mutace zajišťuje, že populace se nestane jednotvárnou a neschopnou dalšího vývoje. Může se stát, že po několika generacích bude díky selekci na jedné pozici každého chromozómu stejný bit. Křížením získáme na této pozici opět stejný bit. Jediný způsob, jak předejít této nežádoucí situaci, je mutace tohoto bitu.

Je vhodné zvolit velmi malou pravděpodobnost mutace (0.5% - 5%). Mutace má samozřejmě stejně jako křížení destruktivní vliv na schémata (mění oblast prohledávání) a z tohoto pohledu je vnímána jako *"nutné zlo"*. Na druhou stranu, některé algoritmy využívající právě mutace a žádného křížení, poskytují velmi komplexní prohledávání s dobrými výsledky. V každém případě je spíše lepší hledat rovnováhu mezi prováděním obou operací, než jednu úplně vynechat.

Před mutací: Po mutaci:

9 Elitářství

Elitářství je metoda umožňující za všech okolností zachovat kvalitní jedince v populaci. Definovaný počet nejlepších jedinců, často jeden, je zachován do další generace v nezměněné podobě. Elitářství slouží jako doplněk selekce, který zabraňuje ztrátě nejlepších jedinců, kteří by nemuseli být vybráni do další generace, popřípadě by mohli být poškozeni destruktivním vlivem křížení a mutace. Použití elitářství výrazně vylepšuje běh algoritmu. V praxi se často používá jeden elitní jedinec, který zabrání tomu, aby v dalším běhu nedošlo ke snížení nejlepšího fitness v populaci.

Kromě elitářství se používá i metoda odstranění nejhoršího jedince z populace (genitor). Tuto metodu se doporučuje provádět ve větších populacích.

10 Škálování

V průběhu genetického algoritmu se zásadně mění rozprostření hodnot fitness mezi jedinci. Na počátku může mít jeden jedinec výrazně větší fitness než ostatní (to vede k předčasné konvergenci) a na konci běhu bývají zase hodnoty fitness všech jedinců velmi blízké. Proto je vhodné nějakým způsobem zajistit udržení selekčního tlaku přibližně na stejné úrovni. To je cílem metody zvané škálování. V této části se zaměříme na použití lineárního škálování.

Při použití **lineárního škálování** chceme škálovat fitness každého jedince v populaci tak, aby škálovaná hodnota fitness byla lineárně svázaná s neškálovanou hodnotou. Škálovanou hodnotu fitness označíme f'. Původní hodnotu f.

$$f' = a \cdot f + b$$
$$f_{\text{avg}} = f'_{\text{avg}}$$
$$f_{\max} = C \cdot f_{\text{avg}}$$

Koeficienty a, b zvolíme tak, aby průměr škálovaných a neškálovaných hodnot fitness byl stejný. Tím zajistíme, že očekávaný počet potomků průměrného jedince v další generaci se nezmění. Další požadavek je, aby škálovaná hodnota byla C násobkem průměrné fitness f_{avg} v generaci. Volbou této konstanty ovlivňujeme velikost rozdílu fitness jedinců v populaci. Volbou $C < 1$ předejdeme prvně zmíněnému problému velmi nadprůměrného řetězce na počátku běhu, snížíme rozdíl mezi hodnotami fitness všech jedinců. Volbou $C > 1$ naopak rozdíl zvýšíme. To se hodí spíše ke konci běhu algoritmu. Kombinací předchozích vztahů dostaneme rovnice pro výpočet koeficientů a a b.

$$a = f_{\text{avg}} \cdot \frac{f_{\max} - C \cdot f_{\text{avg}}}{f_{\max} - f_{\text{avg}}}$$

$$b = \frac{f_{\text{avg}} \cdot (C-1)}{f_{\max} - f_{\text{avg}}}$$

Podobného mechanismu využívá také rank ruleta. Při přidělení ranků každému jedinci snižuje rozdíl mezi hodnotami fitness na 1. Pokud používáme selekci turnajem, nemá užití škálování význam, protože porovnáváme pouze, zda jeden jedinec má větší fitness než jiný a to se provedením škálování nezmění. Další jednoduchá varianta, která se snaží snížit fitnessy jedinců v populaci, je pouhé odečtení nejnižší hodnoty fitness v populaci od každého jedince.

11 Populace

Genetické algoritmy používají k prohledávání mnoho jedinců, kteří tvoří populaci. Populace jedinců zabraňuje uváznutí v lokálním optimu.

Základním problémem, se kterým se musíme potýkat při práci s genetickým algoritmem, je nutnost kódování potenciálních řešení. Pro každý problém se hodí jiný typ kódování. Nejčastěji se používá kódování binárními řetězci, celočíselné a kódování pomocí stromu.

Volba velikosti populace je jeden z faktorů ovlivňujících schopnost algoritmu rychle konvergovat k optimálnímu řešení. Příliš malá populace může způsobit, že algoritmus neprohledá dostatečnou část prohledávaného prostoru, aby našel ucházející řešení. Volba příliš velké populace naopak způsobí zbytečné prodloužení výpočtu, bez efektu na kvalitu nalezeného řešení.

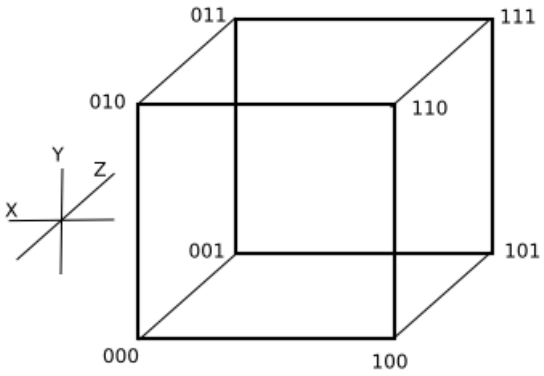
Optimální velikost populace se liší pro různé problémy a především pro zadání "různých velikostí".

12 Matematický popis genetických algoritmů, teorém o schématech

V následujícím textu si ukážeme způsob, jakým se matematicky popisují genetické algoritmy. Nejedná se ovšem o přesný model popisující běh genetického algoritmu. Na závěr jsem uvedl některé nedostatky tohoto modelu. Dále budeme předpokládat pouze jednoduchý genetický algoritmus s binárním kódováním.

K popisu používáme pojem schéma. Schématem nazýváme řetězec stejné délky jako je řetězec v generaci, který kromě znaků 0 a 1 obsahuje znak * sloužící jako zástupný symbol. Řekneme, že řetězec se s daným schématem shoduje, pokud se shodují znaky na pozicích neobsahující znak *.

Pracujeme-li s řetězci délky N, můžeme řetězce mapovat na hrany N-rozměrné krychle. Pro jednoduchost uvedu příklad s řetězci délky 3. Každý takový řetězec představuje souřadnice jedné hrany krychle v prostoru. Schémata pak odpovídají hranám krychle. Každé hraně odpovídá jedno potenciální řešení.



Obrázek 3. Grafické znázornění schémat

Přední stěna krychle odpovídá schématu **0. Pokud bychom chtěli vytvořit čtyřrozměrnou krychli, vytvoříme jí vložením 3-rozměrné krychle do jiné 3-rozměrné krychle a propojíme odpovídající hrany. Hrany vnější krychle očíslujeme stejně jako odpovídající hrany vnitřní krychle. Nakonec doplníme očíslování o čtvrtý bit tak, že ke hranám vnější krychle přidáme 0 a ke hranám vnitřní ktychle 1 na místo prvního bitu. Po této úpravě reprezentuje vnitřní krychli schéma 1*** a vnější krychli 0***.

Zkoumejme nyní, co se bude dít s počtem řetězců shodujících se s daným schématem při různých operacích genetického algoritmu.

12.1 Selekce

Počet řetězců shodujících se s daným schématem v t-tém kroku genetického algoritmu označujeme jako $M(H,t)$. $M(H,t+\text{selekce})$ označuje generaci po provedení selekce, avšak před provedením křížení a mutace.

Uvažujme selekci založenou na principu vážené rulety. Průměr kvalit jedinců, shodujících se se schématem H v t-té generaci $f(H,t)$ určíme jako podíl součtu hodnot fitness jedinců shodujících se se schématem a jejich počtu. Platí tedy: $f(H,t) = \frac{\sum f(H_i,t)}{M(H,t)}$. Podle tohoto vztahu můžeme odhadnout očekávaný počet řetězců shodujících se se schématem při přechodu do následující generace.

$$M(H,t+\text{selekce}) = M(H,t) \cdot \frac{f(H,t)}{\sum f_i} \cdot n \ = M(H,t) \cdot \frac{f(H,t)}{\bar{f}}.$$

Vidíme, že pomocí selekce upřednostňujeme schémata s nadprůměrnou kvalitou, nezískáme ovšem nové vzorky prohledávaného prostoru.

12.2 Křížení

K demonstraci vlivu křížení na schémata použijeme jednoduché křížení, které provádíme rozdělením řetězců v jediném bodě a vzájemnou výměnou informací.

Operace křížení má destruktivní vliv na schémata. To znamená, že při křížení dvou řetězců, z nichž se alespoň jeden neshoduje s daným schématem, můžeme získat řetězce, z nichž se ani jeden s původním schématem neshoduje. Například pro schéma *00*1***1:

Původní řetězce		Po křížení	
1001111	11	1010100	11
1010100	00	1001111	00

Délku schématu H označme $\Delta(H)$. Délka schématu udává vzdálenost mezi první a poslední určenou pozicí schématu. Například *1**0*10*1 má délku 8.

Označíme-li délku řetězce L, může ke křížení dojít na L-1 pozicích v řetězci, přitom ke ztrátě může dojít na $\Delta(H)$ pozicích, neboť musí být na obou stranách od vybrané pozice křížení určená pozice schématu. Pokud se navíc řetězec neshodující se se schématem liší na obou stranách od bodu křížení v alespoň jedné určené pozici, pak se ani jeden výsledný řetězec nebude shodovat s původním schématem, jak ukazuje předešlý příklad.

Né vždy je ale destrukce takto absolutní, může se stát, že alespoň jeden jedinec se po křížení se schématem bude shodovat. Dokonce je možný vznik nového jedince, který se shoduje se schématem i přesto, že ani jeden z rodičů se se schématem neshodoval, např. mějme schéma **001**0 a rodiče 001|01000, 000|10000. Vzniklí potomci jsou 00001000 a 00110000. První potomek se se schématem shoduje a druhý nikoliv, přestože ani jeden rodič se se schématem neshoduje. Tyto zisky zanedbáváme, uvažujeme pouze ztráty a předpokládáme, že pokud budeme křížit na pozici schématu, od které je na obou stranách určená pozice, dojde ke ztrátě. Pokud se oba rodiče se schématem shodují, k destrukci nedojde.

Pravděpodobnost p_d destrukce můžeme vyjádřit takto:

$$\frac{\Delta(H)}{L-1} \cdot (1 - \frac{M(H,t)}{N} \cdot \frac{f(H,t)}{\bar{f}})$$

N ve vztahu označuje velikost populace. $(1 - \frac{M(H,t)}{N} \cdot \frac{f(H,t)}{\bar{f}})$ odpovídá pravděpodobnosti, že náhodně vylosovaný jedinec z populace po selekci se nebude se schématem shodovat. Označme $\frac{\mathrm{M(H,t)}}{\mathrm{N}}$ jako P(H,t). Dále označme pravděpodobnost křížení p_c. Očekávaný počet jedinců shodujících se se schématem po operaci selekce a křížení určíme takto:

$$M(H, t + \text{selekce} + \text{krizeni}) \geq (1 - p_c) \cdot M(H,t) \cdot \frac{f(H,t)}{\bar{f}} + p_c \cdot \left[M(H,t) \cdot \frac{f(H,t)}{\bar{f}} \cdot (1 - p_d) \right]$$

Po úpravě:

$$P(H, t + \text{selekce} + \text{krizeni}) \geq P(H,t) \cdot \frac{f(H,t)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\Delta(H)}{L-1} \cdot (1 - P(H,t) \cdot \frac{f(H,t)}{\bar{f}}) \right]$$

Trochu méně destruktivní charakter na schémata má křížení ve dvou bodech. Uniformní křížení má destruktivnější efekt na schémata, než křížení v jednom bodě. Pro schémata 3. řádu je ve všech ohledech destruktivnější, než křížení ve dvou bodech *(DeJong)*. Přes tyto nedostatky se ukazuje, že uniformní křížení poskytuje velmi dobré výsledky. Také křížení ve více než dvou bodech má velmi destruktivní účinky. Operátor křížení se ale zdá být mnohem mocnějším nástrojem pro generování schémat než operátor mutace.

Mutace má také destruktivní vliv na schémata. Jako taková má zabránit jednotvárnosti populace.

Řád $o(H)$ schématu označuje počet určených pozic ve schématu. Čím více je ve schématu neurčených pozic, tím větší část N-rozměrné krychle je tímto schématem pokryta.

Pravděpodobnost, že na jedné pozici řetězce nedojde ke změně, určíme jako $1 - p_m$, kde p_m označuje pravděpodobnost mutace. Pravděpodobnost, že v celém řetězci nedojde k mutaci, pak odpovídá výrazu $(1 - p_m)^{o(H)}$.

Výsledný vztah vyjadřující teorém o schématech:

$$P(H, t+1) \geq P(H, t) \cdot \frac{f(H,t)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\Delta(H)}{L-1} \cdot (1 - P(H, t) \cdot \frac{f(H,t)}{\bar{f}})\right] \cdot (1\text{-}\mathrm{p_m})^{\mathrm{o(H)}}$$

12.3 Nárůst počtu řetězců

Uvažujme nyní nárůst jedinců shodujících se se schématem před provedením křížení a mutace. Platí tedy $M(H, t + \text{selekce}) = M(H, t) \cdot \frac{f(H,t)}{\bar{f}}$. Dále nechť $f(H, t) = \bar{f} + \bar{f} \cdot c$, kde c je kladná konstanta.

$$M(H, t + \text{selekce}) = M(H, t) \cdot \frac{\bar{f} + \bar{f} \cdot c}{\bar{f}} = M(H, t) \cdot (1 + c).$$

Tento vztah můžeme zobecnit pro t-tou generaci následovně: $M(H, t) = M(H, 0) \cdot (1 + c)^t$. Výsledný vztah ukazuje exponenciální nárůst řetězců ve schématu, jehož kvalita je nadprůměrná. Obdobné pozorování platí pro pokles podprůměrných jedinců.

Ve skutečnosti nemůžeme předvídat počet řetězců shodujících se se schématem o více než jednu generaci dopředu. Exponenciální nárůst počtu jedinců v konečné populaci není možný, protože nárůst řetězců shodujících se se schématem způsobuje snížení jeho relativní kvality vůči průměru. Pokud se se schématem shodují všechny řetězce, bude mít právě průměrnou kvalitu všech řetězců v populaci. Navíc nebereme v úvahu to, co se děje s ostatními schématy (bereme v úvahu pouze jedno schéma odděleně od ostatních). Přes tyto nevýhody si můžeme udělat přibližnou představu o nárůstu jedinců v nadprůměrném schématu i když nepostihujeme celou komplexnost genetických algoritmů. Další informace naleznete např. v [PET], [WHIT].

Nyní můžeme vyslovit **teorém o schématech**:

Teorém 2. *Počet krátkých nadprůměrných schémat nízkého řádu v jednotlivých generacích exponenciálně roste. Tato schémata nazýváme stavebními kameny (building blocks).*

13 Implicitní paralelismus

Každý jedinec v populaci se shoduje právě s $2^L - 1$ schématy, kde L je délka binárního řetězce daného jedince. Jinými slovy, na libovolné pozici daného řetězce může být buď původní symbol nebo zástupný symbol *. Celkem existuje $3^L - 1$ schémat v celém prohledávaném prostoru. Každý jedinec se tedy shoduje s mnoha schématy, především se schématy s nízkým řádem *(tzv. stavební bloky)*.

Tím, že provedeme ohodnocení nějakého řetězce, určujeme také částečně kvalitu schématu, se kterým se daný řetězec shoduje. Pouze jeden řetězec nám moc neřekne, proto je důležité pracovat s celou populací řetězců.

Terorie říká, že pomocí reprodukce a rekombinace se řetězce ve schématech objevují nebo mizí v závislosti na tom, jak jsou kvalitní. Nárůst jedinců s nadprůměrnou kvalitou je exponenciální, stejně tak i pokles jedinců ve schématu s podprůměrnou kvalitou. Tím algoritmus prohledává více ty části prohledávaného prostoru, které obsahují potenciálně lepší řešení.

John Holland dokázal, že počet schémat, která nejsou ničena rekombinací *(tzv. efektivní schémata)* v populaci n jedinců je $\boldsymbol{o}(n^3)$. To znamená, že v generaci se nachází n^3 stavebních bloků. Navzdory tomu, že genetický algoritmus pracuje pouze s n jedinci, pracuje alespoň s n^3 efektivními schématy. Tím dokáže prohledat mnohem více částí prohledávaného prostoru, než jaká je velikost populace (vzorků). Tato vlastnost genetických algoritmů se nazývá implicitní paralelismus. Jedná se o velmi důležitou vlastnost genetických algoritmů, na jejímž základě se vysvětluje jejich funkce.

14 Parametry genetického algoritmu

Prarametry genetického algoritmu rozumíme hodnoty a metody ovlivňující běh algoritmu. Základními parametry jsou: pravděpodobnost křížení, pravděpodobnost mutace, velikost populace, počet kroků algoritmu, metoda křížení, metoda selekce a metoda mutace. Tyto parametry jsou navzájem provázané a není proto možné určit jeden z nich nezávisle na jiném. To znamená, že empiricky získaná "dobrá" velikost populace může být nevýhodná, změníme-li např. pravděpodobnost křížení. Jedním z cílů této práce je nalézt vhodné parametry algoritmu pro řešení vybraných problémů.

15 Metody lokálního prohledávání

Lokální metody mají obecně tendenci nalézat pouze lokální optimum. Naproti tomu metody globálního prohledávání, jako např. genetické algoritmy, se snaží nalézt globální optimum. Obecně běh lokálních algoritmů začíná v nějakém počátečním stavu a v průběhu algoritmu se přesouvá do sousedních stavů, dokud nenalezne řešení (pro případ rozhodovacích problémů) nebo pro optimalizační problémy dostatečně dobré řešení. Prohledávání může skončit také, pokud jsou vyčerpány dostupné technické prostředky (např. paměť). Další metody naleznete v [SIP].

15.1 Simulované ochlazování

Simulované ochlazování *(simulated annealing)* je lokální optimalizační metoda narozdíl od genetického algoritmu. Název pochází z metalurgické metody, při které je kov ohříván a poté postupně kontrolovaně ochlazován. Zvýšením teploty se atomy začnou uvolňovat a náhodně pohybovat ve vyšších energetických hladinách. Při pomalém ochlazování atomy zaujmou s velkou pravděpodobností pozice s nižší energií, než měly před zahřátím.

Stejný proces lze aplikovat v optimalizačních algoritmech. Začínáme s nějakým náhodným stavem. Lokální krok je proveden, vede-li ke zlepšení stavu. Ve speciálním případě je přijat i krok, který vede ke zhoršení stavu. K přijetí zhoršujícího kroku přistupujeme s pravděpodobností, která je odvozena od globálního parametru, který nazýváme teplota. Teplota je postupně snižována, tím se snižuje pravděpodobnost zhoršení stavu a postupně dochází k stabilizaci.

Algoritmus simulated annealing:

```
s = náhodné validní řešení;
nejlepsi = s;
T = InitT();
while(T < frozen()) {
  chyby = 0;
  while(chyby < pocet_tahu) {
    dalsi_stav = sousedni stav stavu s;
    if(evaluation(dalsi_stav) < evaluation(s)) {
      s = dalsi_stav;
      if(evaluation(s) < evaluation(nejlepsi)) nejlepsi = s;
    } else {
      p = random <0,1);
      if(p < e^((evaluation(s) - evaluation(dalsi_stav))/T)) {
        s = dalsi_stav;
      } else {
        chyby++;
      }
    }
  }
  T = 0.8*T;
}
return nejlepsi;
```

15.2 Tabu prohledávání

Tabu prohledávání *(tabu search)* je optimalizační metoda lokálního prohledávání *(local search)*. Tato metoda používá jeden či více tabu seznamů *(tabu lists)* k zamezení cyklení a uváznutí v lokálním minimu. Tabu listy tvoří tabu paměť *(tabu memory)*. Tato paměť obsahuje záznamy o konfiguracích, které byly již navštíveny. Z jednotlivých konfigurací si udržuje pouze příslušné atributy a jejich hodnoty, které slouží jako charakteristika konfigurace. Uložené konfigurace se stávají po určitý počet iterací tabu. Někdy může být tabu omezení příliš přísné, proto za splnění aspiračních podmínek *(aspiration criteria)* může být příslušná konfigurace považována za možný tah přesto, že je tabu. Takovému kritériu může vyhovovat například tah, jehož provedením získáme lepší řešení, než jsme doposud nalezli, tah, který nebyl proveden po definovaný počet iterací nebo jsou-li všechny možné tahy tabu, vybereme některý z nich.

Algoritmus tabu-search:

```
s = náhodné řešení v lokálním minimu;
iterace = 0;
init(tabu_list);
nejlepsi = s;
while(evaluation(s) > 0 AND iterace <  pocet_iteraci) {
   T = urci_tah(moves);  // Vyber nejlepší tah T mezi možnými tahy,
//které nejsou v tabu seznamu a nebo splňují aspirační kritérium.
   proved(T);     // proveď T
   pridej_do_tabu(T); // přidej T do tabu seznamu, odstraň nejstarší
//položku tabu seznamu.
   if(evaluation(s)<evaluation(nejlepsi)) nejlepsi = s;
   iterace++;
}
return nejlepsi;
```

15.3 Hill-Climbing

Hill-Climbing je dalším velmi známým algoritmem lokálního prohledávání. Algoritmus začíná prohledávání v náhodně generovaném stavu. V následujícím kroku se posune do sousedního stavu s nejlepším ohodnocením. Jestliže je dosaženo striktního lokálního minima (stav, kdy každý ze sousedů má vyšší ohodnocení, než je ohodnocení daného stavu), je algoritmus znovu spuštěn v jiném, náhodně vygenerovaném stavu. Max_moves udává maximální počet tahů mezi restarty. Algoritmus musí ohodnotit všechny sousední stavy, než se rozhodne kudy pokračovat. To může ovšem trvat poměrně dlouho.

Algoritmus hill-climbing:

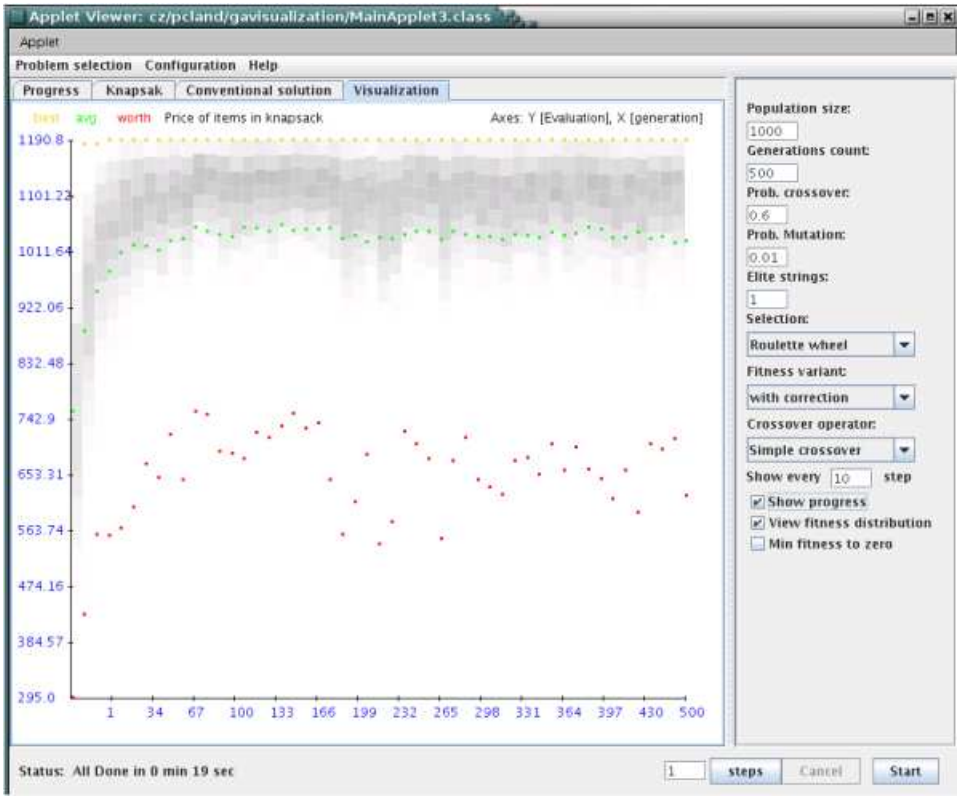
```
s = náhodný stav;

for(j=1;j<=pocet_tahu;j++) {
    if(evaluation(s)==0) {
       return s;
    }
    if(s == striktní lokální minimum) {
      restart();
    } else {
      s = sousední stav s nejmenším ohodnocením;
    }
}
restart();
```

16 Vlastní implementace programu

Testovací program, který je součástí této práce, je navržen pro vizualizaci jednoduchého genetického algoritmu tak, jak je popsán ve čtvrté kapitole. V současné době je implementován problém batohu, obchodního cestujícího, 3-SAT a maximalizace funkcí. V následujícím textu si jednotlivé úlohy popíšeme a uvedeme vhodné, experimentálně získané, parametry genetického algoritmu.

Program byl navržen především pro snadnou rozšiřitelnost. Proto není problém vytvářet vlastní genetické operátory pouze implementací příslušného rozhraní. Obdobně lze přizpůsobit hodnotící funkci. Tato vlastnost programu dává uživateli možnost experimentovat s vlastnoručně vytvořenými operátory.



Obrázek 4. Ukázka GUI

Program byl vytvořen ve vývojovém prostředí NetBeans 5 pod Java 2 Standard Edition (J2SE) 5.0. U starších verzí Javy se mohou vyskytnout problémy, neboť vzájemná kompatibilita nebyla dostatečně testována. Pro toto prostředí jsem se rozhodl především proto, že je Open Source narozdíl například od alternativního Borland JBuilderu. Osobně preferuji svobodný software a software s otevřenými zdrojovými kódy, a proto jsem i vývoj prováděl pod svobodným operačním systémem GNU/Linux.

Důležité datové struktury

Celý program obsahuje více než 100 souborů se zdrojovým kódem. Proto zde uvádím pouze nejdůležitější datové struktury, jejichž správná interpretace je nutnou podmínkou k provádění modifikací a implementaci vlastních genetických operátorů.

Třída Population: Zde se nachází vlastní implementace genetického algoritmu. V této třídě probíhá inicializace algoritmu, přechod z jedné generace do druhé (selekce, křížení, mutace) a vyhodnocení populace. Implementace se drží schématu, který byl zaveden v kapitole 4.

Třída BaseIndividual: Je předchůdcem tříd BinaryIndividual a IntegerIndividual. Obě reprezentují jedince z kterých je složena populace. Třída BinaryIndividual zastřešuje jedince kódované binárními řetězci. Ti jsou reprezentováni polem boolean hodnot. Pokud bychom se rozhodli otestovat reprezentaci např. pomocí spojového seznamu, není třeba v programu nic upravovat, kromě vlastní reprezentace v třídě BinaryIndividual. Podobně v třídě IntegerIndividual se nachází data pro celočíselné, přesněji řečeno, permutační kódování. Jednotlivé permutace jsou uloženy v poli hodnot typu integer. Oba potomci dědí ze svého předka metodu *void mutate(double prob)*, pomocí které je prováděna mutace.

Rozhraní Evaluator: Obsahuje především metodu *double getEvaluation(BaseIndividual bi)*, která vrací ohodnocení jedince. Každá třída poskytující hodnocení implementuje toto rozhraní. Po vytvoření hodnotící třídy jí stačí přidat do seznamu hodnotících tříd daného problému.

Rozhraní SelectionOperator: Třída poskytující služby selekce implementuje toto rozhraní. Nejdůležitější metodou je metoda *BaseIndividual select(BaseIndividual[] population)*, která z pole jedinců vybere jednoho jedince. Následně na něj vrátí referenci. Selekce probíhá stejně u všech problémů, proto stačí nově vytvořený operátor umístit do globálního seznamu selekčních operátorů.

Rozhraní CrossoverOperator: Třída, která implementuje toto rozhraní, musí definovat metodu *BaseIndividual[] crossOver(BaseIndividual, BaseIndividual, strLen)*, která provede vlastní křížení a vrátí seznam nově vytvořených jedinců.

Rozhraní Loadable: Aby bylo možné načítat instance jednotlivých problémů, testovací skripty a jiná data, je nutné umožnit aplikaci načítat data ze souborů. To není z bezpečnostních důvodů apletu povoleno. Proto jsem přistoupil k možnosti načítat data přímo z textových komponent v podobě řetězce. Dalším omezením, které znesnadňuje implementaci tohoto způsobu načítání dat, je omezení přístupu k systémové schránce. Nezbývá tak nic jiného, než používat zkratky operačního systému, pomocí nichž je přenášení dat z apletu a do apletu bezpečné.

Třída BaseProblemPanel: Poskytuje základní rozhraní pro implementaci řešeného problému. Především pak spouští běh algoritmu ve vlastním vlákně a v průběhu běhu algoritmu poskytuje informace o běhu a dosavadním řešení. To je poté zobrazeno ve vizualizačním panelu. Samotná třída BaseProblemPanel je potomkem třídy JPanel. Problémy řešené genetickým algoritmem jsou potomky této třídy.

16.1 Testování

Testování je prováděno pomocí testovacích souborů. Tyto soubory obsahují testovací data a příslušné parametry genetického algoritmu. Formát souborů je následující:

```
# Testing script script
# format:
# run {"test name", PROBLEM_NAME { instance data },
# population_size, crossover_prob, mutation_prob, elites,
selection_method ("params"),
# fitness_variant, crossover_variant , view distribution, min fotness to
zero, steps_to_run }
# PROBLEM_NAME = (ks, sat, tsp)
```

Příkazy run se mohou za sebou opakovat. Není nutné opakovat specifikaci instance zadání. Pokud položka "instance_data" obsahuje text "[last]", použije se předchozí zadání. Komentáře jsou uvozeny znakem #. Příklad vstupu může být následující:

```
    run {"KS test 1", ks { data ... }, 100, 0.7, 0.01, 1, "Roulette
wheel", "with correction" ,"Simple crossover"}

    run {"KS test 1", ks { [last] }, 100, 0.7, 0.01, 1, "Tournament
selection" ("4"), "with correction" ,"Simple crossover"}
```

Data instance jsou data obsahující zadání problému. Pro jejich generování stačí použít příslušné generátory náhodných zadání, které jsou implementovány v programu. Zadání instance problému batohu vypadá následovně:

```
<?XML version="1.0" encoding="UTF-8"?>
<knapsack>
<capacity>Kapacita batohu</capacity>
<item>
<price>Cena_poplożky</price>
<capacity>Objem položky</capacity>
</item>
  ...
</knapsack>
```

Zadání instancí SAT problému jsou ve standardizovaném formátu CNF a májí následující formát:

```
c Toto je komentář
c za p následuje formát formule, počet proměnných a počet klauzulí.
c p cnf 2 5
2 1 1 0
-1 -1 2 0
...
2 -2 -2 0
```

Nakonec se podíváme na formát instancí problému obchodního cestujícího.

```
<?version="1.0" encoding="UTF-8"?>
<point><x>10</x><y>45</y></point>
...
<point><x>1</x><y>15</y></point>
```

Hledání maxima funkce není v testování zahrnuto. Je to proto, že se jedná spíše o demonstrační problém. Je spojen s omezením maximálního rozlišení 30 bitů pro diskretizaci definičního oboru, a proto by neposkytl tak zajímavé údaje jako ostatní problémy.

Provedené testy se nacházejí v dodatku A. Kromě zadání testů, jednotlivých parametrů algoritmu a tabulek s výsledky se zde nachází zhodnocení dílčích výsledků. Pokud není uvedeno jinak, byla pro testy užita populace o velikosti 500 jedinců a 5000 generací. Závěry z testů jsou uvedeny dále zvlášť pro každý problém.

Poznámka 3. Testy byly prováděny na třech PC (Intel Celeron M 1500 Mhz, Intel Celeron 2400 Mhz, Intel Pentium D 2.66 Ghz) za běhu dalších aplikací, nelze proto objektivně porovnávat naměřené časy běhu genetického algoritmu v různých testech. Časové údaje jsou pouze orientační a naznačují časovou náročnost jednotlivých testů.

16.2 Problém batohu

Problém batohu *(Knapsack problem)* je optimalizačním kombinatorickým problémem. Navíc je NP-těžkým problémem, nebyl tedy objeven žádný polynomiální algoritmus, který tento problém řeší. Konvenčně ho lze řešit pouze s exponenciální složitostí. Problém se vyskytuje v mnoha variantách, které jsou zadáním velmi podobné základnímu zadání. Zadání je následující:

Do batohu o kapacitě C máme naskládat předměty, každý o daném objemu Ci tak, aby celkový objem předmětů v batohu byl co největší a zároveň nebyla překročena kapacita batohu C.

Problém, který je implementován v programu, je jednoduchou modifikací základního zadání. Upravený problém je rozšířen následovně: Každému předmětu je navíc přiřazena nějaká cena. Otázka zní, které předměty umístit do batohu tak, aby celková cena předmětů v batohu byla co nejvyšší a zároveň objem těchto předmětů nepřesáhl kapacitu batohu. Pokud se ceny předmětů rovnají jejich objemům, je toto zadání ekvivalentní základnímu zadání.

Řešení pomocí genetického algoritmu

Řešení pomocí exponencielního algoritmu se ukazuje nepoužitelné již pro 45 předmětů. Toto množství předmětů vyžaduje testování 2^{45} možných kombinací předmětů. Na průměrném dnešním domácím PC by výpočet trval několik let. V úvahu tedy přichází přibližné řešení pomocí genetického algoritmu. Řešení konvenčním způsobem je možné otestovat v doprovodném programu. Zde je také zobrazena odhadovaná doba výpočtu.

Volba kódování

První problém při programování genetického algoritmu bývá volba vhodného způsobu kódování potenciálního řešení. V případě problému batohu máme dvě základní možnosti: binární a celočíselné.

Binární kódování je jednodušší pro implementaci, avšak přináší problém nepřípustnosti některých řetězců, který spočívá v překročení maximálního objemu předmětů. Celočíselné kódování tento problém řeší dekódováním řetězců v přípustné řešení. Pro jednoduchost a názornost jsem zvolil binární kódování.

Potenciální řešení zakódujeme binárním řetězcem o délce L odpovídající počtu všech předmětů, které můžeme do batohu umístit. Přítomnost hodnoty "1" na I-té pozici v řetězci určuje přítomnost I-tého předmětu v batohu, hodnota "0" na I-té pozici naopak označuje nepřítomnost I-tého předmětu v batohu. Před spuštěním algoritmu musíme vytvořit neměnný očíslovaný seznam předmětů, který bude sloužit k dekódování řetězce.

Volba hodnotící funkce

Mějme řetězec kódující potenciální řešení podle předchozí definice. Jak určíme kvalitu tohoto řešení ? Z definice problému batohu plyne, že řešení R1 je kvalitnější než řešení R2, pokud součet cen předmětů v batohu z řešení R1 je větší než z R2 (pokud jsou řešení přípustná). Kvalitu řetězce potom určíme jako součet cen všech předmětů v batohu.

Korektnost řetězců

Může se stát, že náhodně vygenerovaný řetězec nebude po dekódování korektním řešením dané instance problému. To může nastat, pokud součet objemů všech předmětů v batohu překročí jeho objem.

První možnost, která se nabízí, je snížení kvality takového řetězce na nějakou konstantní hodnotu tak, aby neměl šanci k vylosování do další generace. S tímto řešením ovšem nastanou problémy, pokud máme hodně předmětů a většina nebo dokonce všechny náhodně vygenerované řetězce jsou nepřípustné. Potom bychom losovali všechny jedince do další generace se stejnou pravděpodobností a to nechceme. Přestože všechny řetězce jsou nepřípustné, některé jsou nepřípustné více a jiné méně a to by se mělo odrazit v pravděpodobnosti vylosování. Díky selekci bychom postupně přicházeli na lepší nepřípustná řešení, až bychom nakonec dostali řešení přípustná. Popsaný postup lze provést pomocí tzv. penalizační funkce. Prakticky to znamená odečtení nějaké hodnoty $\mathbb{P}$ od ohodnocení řetězce. $\mathbb{P}$ je tím větší, čím nepřípustnější řetězec bude. Nepřípustnost, potažmo velikost $\mathbb{P}$ si definujeme jako maximální cenu předmětů, které by mohly být v nadbytečném objemu. Označíme-li C_p kapacitu všech předmětů zakódovaných v potenciálním řešení a C kapacitu batohu, můžeme psát:

$$\mathbb{P} = (C_p - C) \cdot \max_{i=1\ldots N} \left(\frac{P_i}{C_i} \right)$$

Penalizujeme-li řetězce tímto způsobem, přežívají nám v populaci i nepřípustné řetězce. Tyto mají ale šanci po provedení křížení nebo mutace vytvořit přípustné a kvalitní řešení.

Další možností vypořádání se s nepřípustnými řetězci je neumožnit jejich vznik. Takto bychom řetězec generovali tak dlouho, dokud by nebyl přípustný. To ale není příliš šťastné řešení s ohledem na to, že nepřípustný řetězec může vzniknout z přípustného také křížením a mutací. Navíc by opakované generovaní mohlo trvat neúnosně dlouho. Proto bude lepší nepřípustný řetězec upravit na přípustný odebráním některých jedniček z takového řetězce. Náhodným odebráním bychom se mohli připravit o potenciálně kvalitní řešení, musíme tedy rozhodnout, které předměty jsou pro další rozvoj neperspektivní. Nejspíše to budou předměty, které mají na jednotkový objem nejnižší cenu. Tyto předměty budeme z řetězce odebírat, dokuď nebude řetězec přípustný.

Obě výše zmíněné metody se dokáží vyrovnat jak s nepřípustnými řetězci v počáteční populaci, tak i s takovými, které vznikly křížením a mutací. V ukázkovém programu si lze obě možnosti vyzkoušet. Lepších výsledků dosahuje ve většině případů druhá varianta s korekcí nepřípustných řetězců podle zmíněné heuristiky.

Křížení a mutace

Protože jsme zvolili binární kódování, bude probíhat křížení a mutace přesně podle standartních definic uvedených v předchozích částech textu. Mutaci můžeme interpretovat jako přidání nebo odebrání předmětu z batohu. Křížení jako výměnu předmětů v určité části batohu mezi dvěma batohy, které do operace vstupují.

Implementace

- KnapSack.java

  Abstrakce batohu. Obsahuje položky KnapSackItem uložené v datové struktuře Vector. Implementuje rozhraní Loadable.

- KnapSackItem.java

  Reprezentace položky batohu. Každá položka má svůj objem a cenu.

- KnapSackPanel.java

  Zdrojové kódy pro vykreslení GUI.

Testy

Přesto, že varianta fitness s korekcí nepřípustných řetězců dává o něco lepší výsledky než penalizace, znemožňuje plné uplatnění genetických operátorů. Korigované řetězce si jsou často podobné, a proto výměnou jejich částí nezískáme nové řešení. Výrazněji se projevila jen operace mutace a metoda selekce. Mutaci je vhodné volit v rozmezí 0.01 - 0.03. Jako selekci je dobré použít turnaj o velikosti 7 - 10 a populaci o velikosti přibližně 500 jedinců.

Varianta s penalizací přinesla o něco zajímavější výsledky. Pro velikost populace 500 je vhodné volit selekci turnajem o velikosti 7 - 10. Pokud chceme udržet selekční tlak, musíme velikost turnaje přizpůsobit velikosti populace. Selekce ruletou se příliš neosvědčila, což připisuji vysokým hodnotám fitness a malému počtu generací. Pravděpodobnost mutace je nejlepší volit 0.01 - 0.02. Nejlepších výsledků dosahovalo jednoduché křížení s pravděpodobností 0.5 - 0.9. Všechny hodnoty platí pro zadání velikosti 200 - 300 předmětů.

16.3 Problém obchodního cestujícího (TSP)

V problému obchodního cestujícího *(TSP, traveling salesman problem)* máme zadánu množinu $\{c_1, c_2, c_3 \ldots c_N\}$ měst a pro každou dvojci různých měst $\{c_i, c_j\}$ máme zadánu vzdálenost $d\{c_i, c_j\}$. Naším úkolem je najít takové uspořádání měst $\boldsymbol{\pi}$, které minimalizuje hodnotu:

$$\sum_{i=1}^{N-1} d(c_{\boldsymbol{\pi}(i)}, c_{\boldsymbol{\pi}(i+1)}) + d(c_{\boldsymbol{\pi}(N)}, c_{\boldsymbol{\pi}(1)})$$

Tato hodnota odpovídá vzdálenosti uražené cesty *(tour length)* obchodního cestujícího, když projde všechna města v pořadí daném permutací. Nakonec se vrátí do počátečního města, to odpovídá vzdálenosti $d(c_{\boldsymbol{\pi}(N)}, c_{\boldsymbol{\pi}(1)})$. Problém, který řešíme, je symetrický TSP, to znamená, že $d\{c_i, c_j\} = d\{c_j, c_i\}$ pro všechna i, j.

Pokud bychom chtěli zkoušet všechny možné cesty, museli bychom vyzkoušet všechny permutace délky N, kde N je počet měst. To představuje otestování N! možností.

V současnosti se podařilo vyřešit zadání několika tisíc až desítek tisíc měst. Například v roce 2004 byla nalezena optimální cesta mezi 24978 švédskými městy. Tato velmi obtížná zadání jsou obsažena v tzv. TSPLIB. Většina zadání z této knihovny je již vyřešena. Řešení pomocí metod lokálního prohledávání můžete nalézt v [JOGE].

TSP je NP-těžký problém, a tak každý algoritmus pro nalezení optimální cesty musí mít čas běhu, v nejhorším případě rostoucí rychleji, než polynomiálně (pokud platí $P \neq N$). Proto nám nezbývá hledat heuristiky, které pomohou rychle najít skoro optimální řešení nebo využívat nějakého optimalizačního algoritmu, jako například genetického.

Jak bylo řečeno, TSP je optimalizační NP-těžký problém. Hledá cestu minimální délky. TSP-K je NP-úplný. Jedná se o rozhodovací problém s následujícím zadáním: Existuje mezi městy cesta o délce rovné K ?

Volba kódování

Základním požadavkem na kódování je reprezentace zadané cesty v grafu pomocí vhodné datové struktury. K reprezentaci lze použít klasických metod reprezentace grafu, tj. matici sousednosti nebo incidence.

Tyto reprezentace možných řešení jsou příliš komplexní s ohledem na jednoduchost přípustných řešení. Ve skutečnosti bude stačit uložit si posloupnost (permutaci) celých čísel, které reprezentují pořadí navštívených měst (např. 2 5 6 4 1 3). Tento přístup vyžaduje vytvoření speciálních operátorů křížení a mutace.

Hodnotící funkce

Hodnotící funkce je velmi jednoduchá. Kvalitu řešení f odvodíme od délky cesty d. Tu spočteme následovně:

$$d = \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2}, \quad f = \max(d) - d,$$

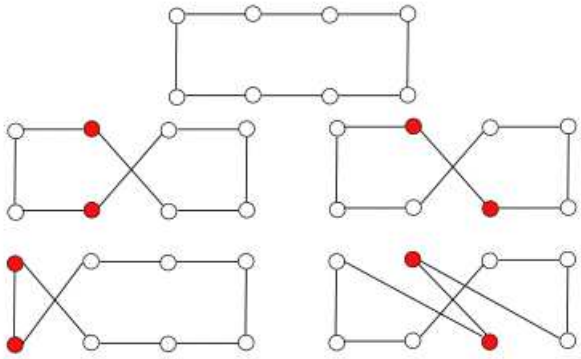
kde n je počet měst, x_i a y_i jsou sořadnice i-tého města. Řetězec reprezentující nejdelší cestu bude mít fitness 0. Tato hodnotící funkce je výpočetně poměrně náročná, což velmi zpomaluje běh algoritmu. Proto je dobré, předpočítat si vzdálenosti mezi městy.

Křížení a mutace

Nejobtížnější část genetického algoritmu řešící TSP je implementace operátorů mutace a křížení. V obou případech musíme zajistit, že výsledkem bude opět přípustná cesta, to znamená, že se nebude v řetězci žádné číslo (město) opakovat.

Nejjednodušším řešením mutace je pouhé přehození dvou náhodně vybraných měst v řetězci. K mutaci lze použít "hladové" přehození (greedy-swap mutace), které provede přehození pouze v případě, že nové řešení bude lepší než řešení před provedením mutace. Tento způsob se mi ovšem příliš neosvědčil, algoritmus většinou uvázl v řešení, které se optimálnímu příliš neblížilo. Je to proto, že pro provedení vylepšení by bylo potřeba provést několik zhoršujících mutací.

Mutace přehozením má následující efekt:



Obrázek 5. Efekt mutace přehozením

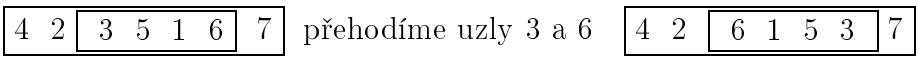
V horní části obrázku se nachází cílový stav. Dále je naznačeno, jak bude probíhat mutace, dojde-li k přehození zvýrazněných uzlů. Přehozením dojde k tomu, že přehozené uzly si vymění sousedy.

4	2	3	5	1	6	7	přehodíme uzly 3 a 6	4	2	6	5	1	3	7

Grafická interpretace může být následující: Představte si na chvíli, že hrany v grafu jsou gumové. Pokud dojde k mutaci, uzly si navzájem prohodí pozice v řetězci, to se ale projeví pouze změnou sousedů obou uzlů. Výsledek je stejný, jako kdybychom prohodili uzly v grafu a ponechali jim původní hrany.

Hrany 2-3, 3-5, 1-6, 6-7 byly nahrazeny hranami 2-6, 6-5, 1-3, 3-7, mutace celkem ovlivní čtyři hrany (pokud aplikujeme metodu zlepšujících mutací, ponecháme novou cestu pouze pokud součet délek nových hran je menší než součet délek původních hran, platí tedy: $\|2\,3\| + \|3\,5\| + \|1\,6\| + \|6\,7\| > \|2\,6\| + \|6\,5\| + \|1\,3\| + \|3\,7\|$).

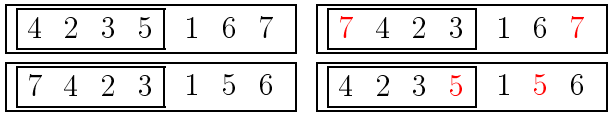
Tento způsob mutace je v praxi velmi používaný, avšak při testování se neprojevoval jako ideální. Přesto, že teoreticky můžeme dosáhnout libovolné permutace, prakticky mají zhoršující mutace malou šanci přežít a tím znemožní provedení dalších mutací, které by vedly ke zlepšení. Z tohoto důvodu by bylo dobré, vytvořit takovou mutaci, která provede nápravu situace znázorněné na obrázku v 2. řádku během jednoho průběhu. Požadovaného efektu jsem dosáhl změnou pořadí uzlů mezi dvěma náhodně vybranými body.



Místo toho, aby byla vedena hrana z uzlu 2 do uzlu 3, povedeme hranu z uzlu 2 do uzlu 6, dále v obou "grafech" vede hrana z 6 do 1, z 5 do 3 a nakonec se liší koncové hrany 3-7 a 6-7.

Zdá se, že takto modifikovaná mutace dává lepší výsledky. Řešení pak tak často neuvázne ve stavu, kdy by stačilo přehodit dvě hrany k úpravě.

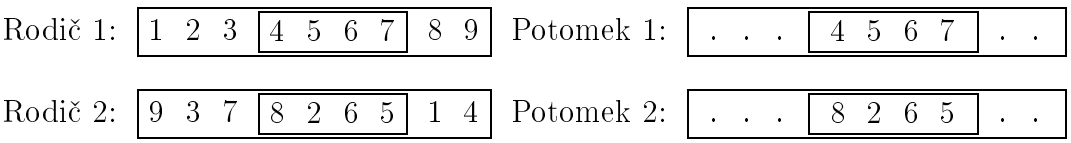
Operace **křížení** bude probíhat poněkud složitěji než obvykle. Aplikováním klasického křížení (ať již v jednom nebo více bodech) bychom mohli získat nepřípustnou cestu. Např.



Abychom zachovali přípustnost řešení, můžeme postupovat následovně:

1. Vybereme část řetězce prvního rodiče.
2. Tuto část zkopírujeme na stejné pozice prvního potomka.
3. Do potomka doplníme zbylá čísla tak, že postupně kopírujeme čísla z druhého řetězce.
   Kopírujeme od poslední určené pozice potomka, pokud narazíme na konec, pokračujme od začátku řetězce.
4. Postup opakujeme pro vytvoření druhého potomka.

Průběh popíšeme na následujícím příkladu:



Nyní zkopírujeme do prvního potomka postupně čísla 1 9 3 8 2 z druhého rodiče a do druhého potomka čísla 9 1 3 4 7 z prvního rodiče.

Rodič 2: 9 3 7 | 8 2 6 5 | 1 4 Potomek 2: 3 4 7 | 8 2 6 5 | 9 1

Tento způsob křížení se označuje jako "order one crossover". Pod tímto názvem je také implementován v testovacím programu.

Další možné řešení je nazývané *"partially mapped crossover"*. Nejprve zavedeme označení R1[i], R2[j] pro hodnotu na i-té pozici prvního rodiče, respektive j-té pozici druhého rodiče. Obdobně P1[i] označuje hodnotu na i-té pozici prvního potomka.

1. Vybereme část prvního rodiče a zkopírujeme ji do prvního potomka.
2. Postupně od prvního bodu křížení procházíme čísla v druhém rodiči, pokud narazíme na číslo R2[i], které dosud není v prvním potomkovi, přesuneme ho podle dalšího bodu do potomka.
3. Podíváme se na číslo P1[i] (=R1[i]) v potomkovi. Víme, že toto číslo nemůžeme zkopírovat z druhého rodiče do potomka (způsobilo by to nepřípustný řetězec). Vyhledáme ho tedy v druhém rodiči a nazveme ho R2[j] (R2[j] = P1[i]). Na pozici potomka P1[j] umístíme R2[i] nalezené v bodu 2. Situace se může zkomplikovat tím, že j padne do rozmezí mezi body křížení. Umístěním R2[i] do potomka na j-tou pozici bychom si nepomohli, protože by stále vznikala kolize při kopírování zbytku z druhého rodiče. Proto najdeme R2[k] tak, aby R1[j] = R2[k]. Pokud je k mimo rozmezí bodů křížení, umístíme R2[i] do P1[k], v opačném případě postup opakujeme.

Situaci objasní příklad:

Rodič 1: 1 2 3 | 4 5 6 7 | 8 9 Potomek 1: . . 2 | 4 5 6 7 | . 8

Rodič 2: 9 3 7 | 8 2 6 5 | 1 4

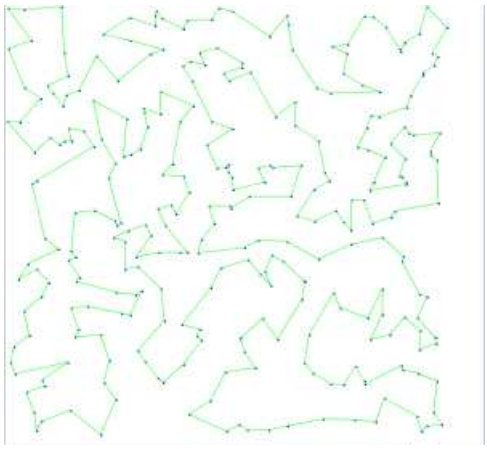
Podle bodu 2 najdeme R2[i] neobsažené v potomkovi. První je hodnota 8. Nyní podle bodu 3 najdeme P1[i] = 4. Tuto hodnotu najdeme v R2 a dostaneme R2[j] = 4, j=9. Na P1[9] umístíme hodnotu R2[i] = 8.

Druhou a poslední hodnotou neobsaženou v potomkovi je 2. Místo ní je v potomkovi číslo 5. Vyhledáme číslo 5 v druhém rodiči a vidíme, že se nachází mezi body křížení. Pokračujeme a místo hodnoty 5 je v potomkovi hodnota 7, tu nalezneme v druhém rodiči a na její místo umístíme do potomka hodnotu 2.

Rodič 1: 1 2 3 | 4 5 6 7 | 8 9 Potomek 1: 9 3 2 | 4 5 6 7 | 1 8

Rodič 2: 9 3 7 | 8 2 6 5 | 1 4 Potomek 2: 1 7 3 | 8 2 6 5 | 4 9

Na následujícím obrázku se nachází nalezené řešení pro instanci 300 náhodně vygenerovaných měst asi po 1/2 hodině výpočtu:



Obrázek 6. Mapa měst

Vidíme, že cesta se nikde nekříží, což je nutná podmínka optimality (vyplývá z trigonometrické nerovnosti). Podmínka to však není postačující a v žádném případě nemáme zaručeno, že nalezená cesta je opravdu optimální. To je dáno tím, že algoritmus se od počátku začne soustředit na jednu variantu, kterou dále optimalizuje. Pokud tato počáteční varianta nevede k optimálnímu řešení, jen těžko provede radikální změnu.

Imlementace

- TSPPanel.java
  Obsahuje definici GUI pro problém TSP.
- TSPInstance.java
  Obsahuje zadání TSP. Data jsou načtená z uživatelem zadaných nebo náhodně vygenerovaných bodů do datové struktury Vector jako dvojice souřadnic (třída City). Protože výpočet vzdáleností mezi 2 body je poměrně náročná operace (2 x mocnina + odmocnina), jsou vzdálenosti mezi každou dvojicí měst vypočteny při strartu algoritmu. Tím má usnadněnu práci fitness funkce, která nemusí znovu počítat vzdálenosti mezi dvojicemi bodů. Po zavedení této optimalizace se běh programu znatelně zrychlil. Navíc jde o formát, do kterého lze převést data prakticky ze všech formátů zadání TSP. TSPInstance implementuje rozhraní Loadable.
- City.java
  Obsahuje dvojici souřadnic daného města.
- CitiesPanel.java
  Tento panel slouží k zadávání instancí problému. Dovede také vykreslit průběh řešení.
- TSPFitness.java
  Pro svou práci využívá předpočítaných hodnot vzdáleností mezi městy, které si vyžádá při spuštění algoritmu.

Testy

Testy byly provedeny na instancích velikosti 150 - 300 měst. Během nich se opět projevila síla selekce turnajem. Tento způsob dosahoval nejlepších výsledků. Velikost turnaje je opět výhodné zvolit 6 - 10. Ze způsobů křížení je nejvhodnější použít křížení "order 1" s pravděpodobností 0.6 - 1. Dosahuje mnohem lepších výsledků než PMX (partially mapped crossover) a křížení v jednom bodě. Křížení "order–1" zachovává narozdíl od PMX pořadí měst a proto nezpůsobí takovou ztrátu informace. V programu je implementována výše popsaná mutace změnou pořadí mezi dvěma uzly, která dosahovala lepších výsledků. Pravděpodobnost mutace v případě TSP a permutačního kódování musíme volit výše, než v případě binárních řetězců, protože se vztahuje k celému řetězci a né pouze k jednomu genu. Proto nejlépe dopadly testy s pravděpodobností mutace 0.05 - 0.1, vyjímečně i více.

16.4 SAT

SAT je zkratkou pro *"satisfability problem"*, nebo-li splnitelnosti booleovské formule. Základní zadání SAT problému zní následovně: Mějme booleovskou formuli F, existuje takové přiřazení hodnot všem proměnným, aby celá formule F byla pravdivá ? Na začátku si uvedeme některé základná pojmy, které budeme dále v textu používat.

Základní pojmy

Formule je definována následovně:

i. Každá logická proměnná je formule.

ii. Jsou-li X a Y formule, pak $\bar{X}, X \wedge Y, X \vee Y, X \Rightarrow Y, X \Leftrightarrow Y$ jsou formule.

iii. Všechny dobře utvořené formule jsou výsledkem aplikace konečného počtu předchozích dvouch pravidel.

Pravdivostní ohodnocení formule je zobrazení, přiřazující formuli pravdivostní hodnotu podle následujících pravidel:

- $(\bar{\alpha})$ je pravdivé pokud α je nepravdivé.
- $(\alpha \wedge \beta)$ je pravdivé, jsou-li α i β pravdivé.
- $(\alpha \vee \beta)$ je nepravdivé, jsou-li α i β nepravdivé.
- $(\alpha \Rightarrow \beta)$ je nepravdivé, pokud α je pravdivé a β nepravdivé.
- $(\alpha \Leftrightarrow \beta)$ je pravdivé, je-li α i β pravdivé nebo α i β nepravdivé.

Logická proměnná nebo její negace se nazývá literál. Součtový term se skládá z disjunkce konečně mnoha literálů, součinový term z konjunkce konečně mnoha literálů. Součtový term obsahující všechny proměnné se nazývá maxterm, součinový term obsahující všechny proměnné nazýváme minterm. *Příklad maxtermu:* $(a_1 \vee \bar{a_2} \vee ...a_n)$.

CNF je zkratka pro conjuctive normal form *(konjunktivní normálová forma)*. CNF nazýváme maxterm nebo konjunkci konečně mnoha maxtermů. DNF disjuctive normal form *(disjunktivní normálová forma)* označuje minterm nebo disjunkci konečně mnoha mintermů.

Teorém 4. *(Cook-Levin) Problém SAT je NP-úplný.*

Důkaz předcházejícího teorému lze nalést např. v [SIP].

Problém, který řeší doprovodný program, je speciálním případem SAT problému nazývaný 3-SAT. V problému 3-SAT zjišťujeme, zda formule v konjunktivní normálové formě (CNF), jejíž každý maxterm obsahuje právě tři literály, je splnitelná. Na první pohled je zřejmé, že instance toho problému jsou podmnožinou instancí SAT problému (jde o SAT problém s daným omezením). Víme tedy, že pokud je tento problém NP-úplný, pak je i SAT problém NP-úplný. Zatím ale nevíme, zda opravdu NP-úplný je. Třeba je SAT problém tak složitý právě proto, že obsahuje libovolné množství literálů v každé klauzuli. Nyní ukážeme, že tomu tak není a libovolný SAT problém můžeme převést na ekvivalentní 3-SAT problém.

Teorém 5. *3-SAT problém je NP-úplný.*

Důkaz. Abychom teorém dokázali, musíme ukázat následující:

i. 3-SAT leží ve třídě NP. To platí (stačí uhodnout nedeterministickým polynomiálním algoritmem přiřazení hodnot jednotlivým proměnným).

ii. 3-SAT je NP-těžký, to znamená, že každý problém $L \in \mathrm{NP}$ lze redukovat na ekvivalentní 3-SAT problém. To ukážeme existencí polynomiální redukce SAT problému na 3-SAT. Potřebujeme tedy najít polynomiální algoritmus, který převede CNF formuli α na 3-CNF formuli β tak, aby β byla splnitelná právě když α je splnitelná. Každou klauzuli α velikosti > 3 převedeme na novou klauzuli velikosti 3.

Uvažujme klauzuli $C = (l_1 \vee l_2 \vee l_3 \ldots l_m)$. Navíc zavedeme novou proměnnou y a budeme požadovat splnění 2 podmínek:

1. y je logicky ekvivalentní s $(l_1 \vee l_2 \ldots l_{m-2})$. To je možné zapsat následovně:

$$(y \Rightarrow (l_1 \vee l_2 \ldots l_{m-2})) \wedge ((l_1 \vee l_2 \ldots l_{m-2}) \Rightarrow y)$$

2. $(y \vee l_{m-1} \vee l_m)$ je pravdivá.

Původní klauzule je splnitelná, pokud jsou obě podmínky splněny. Nyní použijeme vztah $(a \Rightarrow b) \Leftrightarrow (\bar{a} \vee b)$. První pravidlo přepíšeme následovně:

$$(\bar{y} \vee l_1 \vee l_2 \ldots l_{m-2}) \wedge (\overline{(l_1 \vee l_2 \ldots l_{m-2})} \vee y) \Leftrightarrow (\bar{y} \vee l_1 \vee l_2 \ldots . l_{m-2}) \wedge ((\bar{l}_1 \vee y) \wedge (\bar{l}_2 \vee y) \ldots (\bar{l}_{m-1} \vee y))$$

Po tomto převodu máme k dispozici jednu klauzuli velikosti m-1 a m-2 klauzulí velikosti 2. Klazuli o velikosti m-1 můžeme nyní opět transformovat na konjunkci klauzule velikosti 3, m-3 klauzulí o velikosti 2 a klauzule o velikosti m-2. Takto lze rekurzivně pokračovat, dokud nezískáme konjunkci klauzulí o velikosti maximálně 3. Celkový počet klauzulí bude maximálně roven $o(m^2)$.
Můžeme učinit ještě jednu úpravu algoritmu. Z formule:

$$(y \Rightarrow (l_1 \vee l_2 \ldots l_{m-2})) \wedge ((l_1 \vee l_2 \ldots l_{m-2}) \Rightarrow y)$$

nám stačí zachovat $y \Rightarrow (l_1 \vee l_2 \ldots l_{m-2})$ nebo-li $(\bar{y} \vee l_1 \vee l_2 \ldots l_{m-2})$. Právě když bude splnitelná formule $(\bar{y} \vee l_1 \vee l_2 \ldots l_{m-2}) \wedge (y \vee l_{m-1} \vee l_m)$, bude splnitelná i původní formule C. Pokud je tato formule pravdivá v nějakém ohodnocení, y může být buď PRAVDA nebo NEPRAVDA. V prvním případě, kdy y je pravdivé, musí být jeden z literálů $l_1 \vee l_2 \ldots l_{m-2}$ pravdivý. Pokud je y nepravda, musí být l_{m-1} nebo l_m pravdivé. V obou případech je C také pravdivá, nebo-li celá formule je pravdivá, právě když alespoň jeden z literálů $l_1 \vee l_2 \ldots l_m$ je pravdivý. Opakovaným použitím těchto pravidel získáme rozklad na konjunkci klauzulí o velikosti 3.

Tím jme ukázali, že problém 3-SAT je NP-úplný.

□

Volba kódování

Při řešení si tentokráte vystačíme s binárním kódováním. Jednotlivé bity řetězce budou označovat hodnoty logických proměnných a to tak, že první proměnná odpovídá prvnímu bitu, následující druhému bitu atd. Celková délka řetězce je dána počtem proměnných ve formuli.

Hodnotící funkce

Zadání SAT problému představuje rozhodovací problém. Abychom ho mohli řešit pomocí genetického algoritmu, je nutné definovat odpovídající optimalizační problém. Formule může být splněna nebo nesplněna, naším cílem je nalézt takovou hodnotící funkci (optimalizační kritérium), která by nějakým způsobem podávala informaci o tom, jestli se blížíme splnění a jak moc. Jako ohodnocení zvolíme počet pravdivých klauzulí. Snažíme se tedy optimalizovat řešení tak, aby obsahovalo co nejvíce pravdivých klauzulí. Pokud jsou všechny pravdivé, je pravdivá celá formule a víme, že formule je splnitelná. Implementace fitness funkce v doprovodném programu je velmi podobná. Zde kvalita odpovídá desetinásobku počtu procent splněných klauzulí. To znamená, že maximální hodnota je 1000.

Křížení a mutace

Díky použití binárního kódování není potřeba nějak upravovat současné operátory křížení a mutace pro binární řetězce. Jsou však možná specifická vylepšení právě pro problém SAT, nejsou však v programu implementována. Mutaci interpretujeme jako změnu hodnoty logické proměnné, křížení umožuje předání hodnot logických proměnných mezi jedinci.

Implementace

- SATPanel.java
  Zdrojové kódy pro zobrazení GUI.
- Lexan.java
  Lexikální analyzátor dat v CNF formátu.
- SATGen.java
  Generátor náhodných SAT instancí.
- SATInstance.java
  Abstrakce SAT instance, dovede parsovat data z CNF formátu, implementuje rozhraní Loadable.

Testy

Z testu vyšla nejlépe selekce turnajem o velikosti 7 - 8. Žádný způsob křížení jednoznačně nepřekonal ostatní metody, o něco lépe ale dopadlo uniformní křížení a křížení ve dvou bodech, než jednoduché křížení (v jednom bodě). Ani vhodnou pravděpodobnost křížení nelze jednoznačně říci, protože v jednotlivých testech se podstatně lišila. Vhodná pravděpodobnost mutace je 0.01 - 0.05.

16.5 Maximalizace funkcí

V poslední části se podíváme na problém hledaní maxima funkce na zadaném intervalu. Optimalizace probíhá na základě funkční hodnoty. My hledáme takový bod definičního oboru, pro který je funkční hodnota maximální. Program umožňuje zadání vlastní funkce, na které je demonstrován běh algoritmu.

Nejprve se podívejme, proč vlastně řešit tento problém pomocí genetického algoritmu. Prvním nápadem by mohlo být vyhodnocení každého bodu definičního oboru a nalezení maxima. Tento přístup ovšem není vhodný ze stejných důvodů, jako pro prohledávání všech možností například u problému batohu. Pokud chceme funkci prohledávat dostatečně zevrubně, použijeme mnoho bitů r pro reprezentaci bodu v rozmezí prohledávání. Pokud bychom chtěli prohledat celou funkci, museli bychom projít 2^r možností. Jiná varianta by mohla vypadat následovně: Vybereme náhodně jeden bod definičního oboru a budeme se posouvat do lepšího ze sousedních bodů, dokud se nestane, že oba sousedé budou horší. Toto je přesně metoda lokálního prohledávání a s velkou pravděpodobností bychom skončili jen v lokálním maximu. Můžeme poté algoritmus restartovat v jiném náhodném bodě a porovnat výsledky. Pokud najdeme lepší bod, nahradíme dosavadní řešení tímto bodem a algoritmus můžeme opět restartovat. Jednodušší variantou je použití genetického algoritmu, který svou populací pokryje prohledávaný prostor a bude se stále více zaměřovat na perspektivní části funkce. Obdobná pozorování platí i pro ostatní problémy, které řešíme genetickým algoritmem.

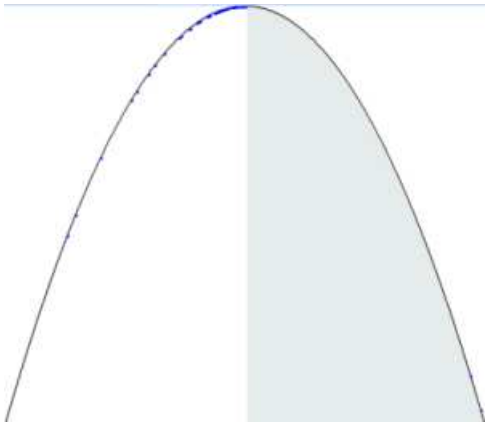
Volba kódování

Na první pohled by se zdálo, že volba kódování je jasná. Použijeme binární kódování, body definičního oboru budeme reprezentovat jako binární čísla. Podívejme se, jak bude probíhat dekódování. První překážkou jsou záporné hodnoty. Dejme tomu, že se snažíme nalézt maximum funkce:

$$f(x) = -(x^2) \text{ na intervalu -10 až 10, rozlišení 20}$$

Záporné hodnoty se vyskytují jak v definičním oboru, tak v oboru funkčních hodnot. Při dekódování se snažíme binární řetězec jednoznačně převést na jeden bod v rozmezí daném intervaly funkce. Máme tedy binární řetězec o délce 20. Mohli bychom například prohlásit, že první bit bude reprezentovat znaménko a zbytek řetězce číslo v intervalu 0 až 10. Převedeme tedy zbylých 19 bitů na desítkový zápis a vynásobíme ho číslem (10/(2^19-1)). Tím dostaneme číslo v intervalu 0 až 10. Pokud by byl interval funkce např. -3 až 7, můžeme mechanismus vylepšit tím, že hodnoty promítneme do intervalu <3,0) a <0,7>. Tím ale preferujeme interval <3,0), kterému přesto že je menší než <0,7>, přidělíme stejný počet bitů. Výše popsaný mechanismus dekódování jsem zavrhl, protože zbytečně přináší spoustu problémů. Dekódování použité v programu funguje následovně: Celých 20 bitů převedeme na desítkové číslo C. Toto číslo promítneme na interval <0,20> a přičteme -10, neboli dekódovanou hodnotu vypočteme takto: (C/(2^20-1))*abs(-10 - 10) + (-10).

Nyní máme dekódován bod, který ovšem nemusí patřit do definičního oboru. Funkce v tomto bodě nemusí být definována. S tímto problémem si ovšem poradí hodnotící funkce. Při testování programu s tímto postupem dekódování marazíme na zvláštní jev, který je naznačen na obrázku:



Obrázek 7. Rozložení populace

Na obrázku vidíme, že jedinci se ke konci běhu algoritmu začali shromažďovat pouze v levé časti průběhu (jedinci jsou vyznačeni tmavě modrými body). Je to dáno tím, že přesně v polovině dochází k velké změně zakódovaných jedinců. Poslední binární číslo před polovinou je 01111 ... 1. První hodnota za polovinou je kódována jako 10000 ... 0. Tyto dvě hodnoty se liší ve všech bitech. Postupem běhu genetického algoritmu se stále více jedinců shromažďuje v jedné z těchto částí. Jediný způsob, jak dosáhnout přechodu, je pak mutace prvního bitu, protože křížením s jedincem který se nachází ve stejné části, dostaneme opět jedince se stejným prvním bitem. Tento stav je jistě nežádoucí, protože algoritmus se zaměřuje pouze na jednu z perspektivních částí prohledávaného prostoru. Alternativně se jedinci mohou shromažďovat na pravé straně.

Zmíněný problém řeší Grayovo kódování. Jeho použitím docílíme, aby malá změna v binárním řetězci vedla k malé změně hodnoty po dekódování. To u binárního kódování neplatí. Jediná změna oproti předchozí verzi bude způsob dekódování binárního řetězce. Rozdíl mezi způsoby dekódování ukazuje následující tabulka:

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Z tabulky je patrné, že dvě sousedící desítková čísla se v Grayově kódu liší pouze v jednom bitu. Po provedení této úpravy můžeme pozorovat rovnoměrné rozložení jedinců i ke konci běhu algoritmu okolo maxima funkce.

Hodnotící funkce

Pro zjištění ohodnocení stačí dosadit dekódovanou hodnotu z jedince do optimalizované funkce a získat tak funkční hodnotu. Problémy přináší opět přítomnost záporných hodnot a navíc fakt, že funkce nemusí být ve všech bodech definována. Záporné hodnoty vyřešíme opět přičtením minimálního fitness v populaci. K tomu ale potřebujeme dva průchody celou populací navíc. První k nalezení minimální hodnoty a druhý k přehodnocení fitness. Alternativně lze přičíst minimum z posledních N generací a pokud bude hodnota stále menší než 0, upravíme ji na 0. Nedefinované hodnoty by neměly mít šanci projít do další generace. Zajistíme to tím, že jim přiřadíme nulovou hodnotu fitness.

Křížení a mutace

Křížení a mutaci provádíme standardními operátory, které jsou popsány výše.

Implementace

- FunctionPanel.java
  Zdrojový kód pro zobrazení GUI.
- GraphPanel.java
  Panel pro vizualizaci běhu algoritmu, zobrazuje graf funkce a navíc pokud má k dispozici populaci, promítne jednotlivé jedince jako body na graf funkce.
- FunctionInstance.java
  Instance funkce. Funkce je uložena jako odkaz na kořen abstraktního syntaktického stromu. Dále obsahuje dolní a horní interval vykreslování a rozlišení, které bude použito pro diskretizaci hodnot v rozmezí prohledávání. Bohužel, zatím je implementována pouze práce s maximálním rozlišením 30, což přináší podstatná omezení.
- Lexan: FunctionToken, InputSymbol, Lexan, LexanInput, NumberToken, StringToken, Token, TokenType, VariableToken.
  Třídy lexikálního analyzátoru.
- Syntaktický analyzátor: Syntax.java, ParseErrorException
  Implementace syntaktického analyzátoru.
- Uzly: AbsNode, AdditionNode, BinaryOperation, CosNode, DivisionNode, FunctionNode, LnNode, LogNode, ModuloNode, MultiplicationNode, Node, NumberNode, PowerNode, Priorities, SinNode, SqrtNode, SubtractionNode, TernaryOperation, TgNode, UnaryMinus, UnaryOperation, UnaryPlus, VariableNode
  Uzly abstraktního syntaktického stromu, používaného k reprezentaci a vyhodnocení výrazu.

17 Závěr

V této práci jsem se zabýval základy genetických algoritmů. Text začíná porovnáním různých skupin výpočetních problémů, mezi které patří optimalizační problémy řešené genetickými algoritmy. Pokračuje výkladem principů a způsobů implementace genetického algoritmu. Součástí práce jsou i praktické příklady aplikace genetického algoritmu na řadě jednoduchých problémů.

Ve všech provedených testech dosahuje nejlepšího výsledku selekce turnajem. Je to částečně způsobeno poměrně malým počtem generací běhu algoritmu. Počet generací byl zvolen 5000 při populaci 500 jedinců, což není optimální hodnota pro všechny parametry algoritmu. Při zvyšování velikosti turnaje zvyšujeme selekční tlak a tak také rychlost konvergence. Algoritmus se začne soustředit velmi rychle na jedno řešení, a proto dokáže rychleji konvergovat.

Prvním řešeným problémem byl problém batohu. Na tomto problému byly ukázány dvě metody práce s nepřípustnými řetězci. Provedené testy naznačují, že nejlepší výsledky s ohledem na časovou náročnost výpočtu při řešení problému batohu s 200 - 300 předměty, získáme použitím selekce turnajem o velikosti 7 - 10, při velikosti populace přibližně 500 jedinců, mutace v rozmezí pravděpodobností 0.01 - 0.03 a křížení v jednom bodě s pravděpodobností 0.5 - 0.9. O něco lepší výsledky získáme použitím varianty fitness s korekcí než s penalizací.

Dalším řešeným problémem je problém obchodního cestujícího. Při jeho řešení bylo použito permutační kódování a speciální způsoby křížení a mutace. V testech pro 150 - 300 měst vychází dobře selekce turnajem o velikosti 6 - 10, křížení "order 1" s pravděpodobností 0.6 - 1 a mutace změnou pořadí mezi dvěma náhodně vylosovanými body s pravděpodobností kolem 0.08.

Dále jsem řešil problém maximalizace funkce na zadaném intervalu, který slouží spíše čistě k vizualizaci běhu algoritmu.

Posledním řešeným problémem byl problém splnitelnosti booleovské formule. Tento problém ukazuje použití genetických algoritmů při řešení rozhodovacího problému převedením na problém optimalizační. Při testování opět nejlépe vycházela selekce turnajem (velikosti 7 - 8). Dobré výsledky získáme užitím mutace s pravděpodobností 0.01 - 0.05. Bohužel se mi nepodařilo určit nejvhodnější způsob křížení, protože všechny metody vycházely velmi podobně.

Myslím si, že vytvořený vizualizační aplet může být přínosnou pomůckou při studiu genetických algoritmů.

18 Použité materiály a literatura

[SIP] Michael Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.

[MIT] Melanie Mitchell, An Introduction to Genetic Algorithms (Complex Adaptive Systems).

[GOLDBERG] Genetic Algorithms in Search, Optimization, and Machine Learning by David E. Goldberg.

[PET] Ivo Peterka, Genetické algoritmy.

[WHIT] Whitely, D., A Genetic Algorithm Tutorial, Colorado State University, 1993.

[BAR] Roman Barták, Guide to Constraint Programming.

[JOGE] David S. Johnson, Lyle A. McGeoch, The Traveling Salesman Problem: A Case Study in Local Optimization.

Dodatek A Testy

A.1 Podmínky testů

Každý test je spouštěn 5 krát. Ve výsledných tabulkách najdete průměr z nejlepších hodnot těchto 5 testů, nejlepší nalezený výsledek a nejhorší nalezený výsledek z těchto 5 testů. Dále pak průměrný počet kroků potřebný na nalezení konečného řešení a celkovou dobu běhu testů. Pokud není stanoveno jinak, genetický algoritmus je spuštěn s následujícími parametry:

Velikost populace: 500, počet kroků (generací): 5000, pravděpodobnost křížení: 0.8, pravděpodobnost mutace: 0.01, 1 elitní jedinec (vždy kopírován do nové generace) a selekce ruletou.

A.2 Testy Batoh

Test 1
Nejprve jsem se rozhodl ověřit schopnost genetického algoritmu nalézt nejlepší řešení. Abych byl schopen nejlepší řešení určit, použil jsem pouze 31 předmětů. Výpočet konvenčním řešením trval téměř 3 hodiny. Testovací sada předmětů je obsažena ve standardním nastavení programu a objeví se po spuštění programu. Pro takto malé množství předmětů jsem zvolil také malé hodnoty velikosti populace a počtu kroků genetického algoritmu. Nastavení je shrnuto v tabulce 1.

Konvenčně vypočtené řešení: **1190,8**

Parametry genetického algoritmu:

prst. křížení	prst. mutace	Elitářů	Kroků	Populace	Selekce	Křížení	Test
0.8	0.01	1	3000	100	Ruleta	2 point	ks_basic.test

Výsledek:

	1.test	2.test	3.test	4.test	5.test
Fitness s korekcí	1190.79	1190.8	1190.8	1190.8	1190.8
Fitness s penalizací	1189.69	1189.69	1189.69	1189.69	1189.69

Tabulka 2. Test 1

Zdá se, že řešení pomocí korekce nepřípustných řetězců přináší lepší výsledky. Počkejme ovšem na další testy, které prověří algoritmus ve spojitosti s jinými operátory a parametry.

Test 2

Nyní se pokusíme určit vhodnou velikost populace. Vhodná velikost populace je samozřejmě závislá na počtu předmětů, které umístíme do batohu. K provádění testů použijeme následující parametry algoritmu:

prst. křížení	prst. mutace	Elitářů	Kroků	Populace	Selekce	Křížení
0.8	0.01	1	5000	testována	Turnaj 4	2 point

Název souboru:	ks1_penalization.test				
Předmětů:	200		Odhad:	4960.1	
Velikost populace	Průměr z 5 testů		Z 5 testů		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
10	4888.32	4763	4908.1	4862.5	16 sec
20	4930.74	3773	4943.1	4910.4	31 sec
40	4933.15	4689	4940.1	4925.4	1 min
80	4934.78	3733	4953.3	4917.1	2 min
100	4938.28	3413	4949.4	4924.7	2 min 30 sec
200	4941.56	4267	4954.0	4925.3	5 min
400	4948.24	2371	4956.0	4943.6	10 min 30 sec
600	4950.2	2625	4959.1	4940.8	17 min
800	4946.0	3029	4955.7	4938.3	22 min 30 sec
1000	4947.68	3541	4957.2	4938.1	27 min 30 sec
2000	4953.02	2033	4960.9	4949.6	56 min 38 sec

Tabulka 3. Test 2

Nejlépe dopadla populace největší o velikosti 2000. Jen o málo hůře pak populace o velikosti 600. Variantu s korekcí nemá cenu uvádět, protože pro všechny velikosti populace dopadla téměř stejně a to včetně malých populací.

Test 3

Tento test by měl vzájemě porovnat metody selekce. Je použita varianta fitness s penalizací.

prst. křížení	prst. mutace	Elitářů	Kroků	Populace	Selekce	Křížení
0.8	0.01	1	5000	500	testována	2 point

Název souboru:	ks_selectionmethods_penalization.test				
Předmětů:	200		Odhad:	4960.1	
Metoda selekce	Průměr z 5 testů		Z 5 testů řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
Roulette	4611.64	4489	4750.4	4440.5	13 min 43 sec
Tournament 2	4816.38	4739	4886.2	4781.1	12 min 55 sec
Tournament 3	4904.3	4183	4926.4	4891.8	13 min 5 sec
Tournament 4	4943.28	3593	4947.0	4941.3	13 min 5 sec
Tournament 5	4960.28	2793	4962.1	4957.3	13 min 5 sec
Tournament 6	4961.92	2041	4964.0	4957.9	13 min 8 sec
Tournament 7	4963.32	1807	4964.0	4962.3	13 min 10 sec
Tournament 8	4962.38	1413	4964.0	4959.7	13 min 10 sec
Tournament 9	4963.34	1253	4963.6	4962.3	13 min 14 sec
Tournament 10	4960.84	1481	4964.0	4957.9	13 min 19 sec
Tournament 20	4960.52	1705	4963.6	4957.8	13 min 36 sec
Rank roulette	4810.0	4207	4830.1	4772.9	14 min 40 sec
Random	4157.98	4741	4346.3	4045.6	13 min 33 sec

Tabulka 4. Test 3, data 1

Název souboru:	ks_selectionmethods_penalization_data2.test				
Předmětů:	250		Odhad:	5593.8	
Metoda selekce	Průměr z 5 testů		Z 5 testů řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
Roulette	4990.52	4393	5097.9	4864.0	16 min 31 sec
Tournament 2	5306.65	4477	5345.1	5226.8	16 min 0 sec
Tournament 3	5514.16	4607	5566.0	5471.9	16 min 11 sec
Tournament 4	5573.94	3645	5595.56	5549.8	16 min 10 sec
Tournament 5	5597.34	3959	5609.5	5583.8	16 min 23 sec
Tournament 6	5604.4	3251	5612.7	5594.5	16 min 27 sec
Tournament 7	5622.26	3081	5629.4	5616.1	16 min 15 sec
Tournament 8	5627.62	3535	5631.5	5625.5	16 min 33 sec
Tournament 9	5632.14	2967	5636.4	5623.2	16 min 25 sec
Tournament 10	5632.14	2967	5636.4	5627.2	16 min 20 sec
Tournament 20	5633.76	1891	5636.4	5628.9	16 min 45 sec
Rank roulette	5320.04	3957	5392.0	5208.2	17 min 45 sec
Random	4505.82	4699	4571.2	4344.4	16 min 11 sec

Tabulka 5. Test 3, data 2

Test dopadl poměrně nečekaně, velmi dobré výsledky dosahují selekce na základě turnaje.

Test 4
V tomto testu vyzkoušíme různé operátory a pravděpodobnost křížení a vliv pravděpodobnosti mutace na nalezené řešení.

prst. křížení	prst. mutace	Elitářů	Kroků	Populace	Selekce	Křížení
testována	0.01	1	5000	500	Ruleta	testováno

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point 0 corr	4963.66	1109.0	4964.0	4963.6	22 min 24 sec
KS cross 1 point 0.1 corr	4963.68	1337.0	4964.0	4963.6	18 min 31 sec
KS cross 1 point 0.2 corr	4963.6	997.0	4963.6	4963.6	19 min 45 sec
KS cross 1 point 0.3 corr	4963.6	603.0	4963.6	4963.6	19 min 52 sec
KS cross 1 point 0.4 corr	4963.68	1703.0	4964.0	4963.6	21 min 8 sec
KS cross 1 point 0.5 corr	4963.6	475.0	4963.6	4963.6	20 min 51 sec
KS cross 1 point 0.6 corr	4963.6	1651.0	4963.6	4963.6	20 min 46 sec
KS cross 1 point 0.7 corr	4963.6	1511.0	4963.6	4963.6	21 min 8 sec
KS cross 1 point 0.8 corr	4963.68	775.0	4964.0	4963.6	19 min 47 sec
KS cross 1 point 0.9 corr	4963.6	1033.0	4963.6	4963.6	22 min 59 sec
KS cross 1 point 1 corr	4963.6	697.0	4963.6	4963.6	24 min 15 sec

Tabulka 6. Test 4

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point 0 corr	4963.59	651.0	4963.6	4963.6	23 min 32 sec
KS cross 2 point 0.1 corr	4963.6	573.0	4963.6	4963.6	22 min 3 sec
KS cross 2 point 0.2 corr	4963.6	1009.0	4963.6	4963.6	21 min 14 sec
KS cross 2 point 0.3 corr	4963.6	1385.0	4963.6	4963.6	23 min 31 sec
KS cross 2 point 0.4 corr	4963.6	1699.0	4963.6	4963.6	22 min 52 sec
KS cross 2 point 0.5 corr	4963.6	1159.0	4963.6	4963.6	24 min 20 sec
KS cross 2 point 0.6 corr	4963.6	1173.0	4963.6	4963.6	24 min 51 sec
KS cross 2 point 0.7 corr	4963.6	799.0	4963.6	4963.6	27 min 10 sec
KS cross 2 point 0.8 corr	4963.6	663.0	4963.6	4963.6	33 min 20 sec
KS cross 2 point 0.9 corr	4963.68	1069.0	4964.0	4963.6	32 min 26 sec
KS cross 2 point 1 corr	4963.68	607.0	4964.0	4963.6	32 min 3 sec

Tabulka 7. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 corr	4963.59	439.0	4963.6	4963.6	22 min 49 sec
KS Uniform crossover 0.1 corr	4963.6	1373.0	4963.6	4963.6	20 min 19 sec
KS Uniform crossover 0.2 corr	4963.6	1631.0	4963.6	4963.6	20 min 27 sec
KS Uniform crossover 0.3 corr	4963.6	873.0	4963.6	4963.6	21 min 17 sec
KS Uniform crossover 0.4 corr	4963.6	449.0	4963.6	4963.6	21 min 16 sec
KS Uniform crossover 0.5 corr	4963.68	1651.0	4964.0	4963.6	21 min 33 sec
KS Uniform crossover 0.6 corr	4963.6	983.0	4963.6	4963.6	21 min 12 sec
KS Uniform crossover 0.7 corr	4963.68	3127.0	4964.0	4963.6	22 min 26 sec
KS Uniform crossover 0.8 corr	4963.24	1557.0	4964.0	4961.4	22 min 23 sec
KS Uniform crossover 0.9 corr	4963.6	615.0	4963.6	4963.6	22 min 32 sec
KS Uniform crossover 1 corr	4963.6	929.0	4963.6	4963.6	24 min 12 sec

Tabulka 8. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point penalization 0	4835.48	4042.0	4923.3	4791.09	23 min 8 sec
KS cross 1 point penalization 0.1	4826.47	4669.0	4848.0	4803.6	20 min 18 sec
KS cross 1 point penalization 0.2	4780.8	4607.0	4821.3	4748.99	19 min 11 sec
KS cross 1 point penalization 0.3	4762.0	4737.0	4811.1	4724.39	19 min 25 sec
KS cross 1 point penalization 0.4	4730.55	4353.0	4779.8	4679.89	26 min 43 sec
KS cross 1 point penalization 0.5	4705.6	4549.0	4752.8	4637.59	30 min 18 sec
KS cross 1 point penalization 0.6	4719.33	4297.0	4790.79	4604.59	20 min 28 sec
KS cross 1 point penalization 0.7	4641.2	4695.0	4707.1	4516.19	21 min 3 sec
KS cross 1 point penalization 0.8	4491.89	4439.0	4568.5	4398.09	21 min 47 sec
KS cross 1 point penalization 0.9	4457.59	4557.0	4569.89	4362.59	22 min 33 sec
KS cross 1 point penalization 1	4290.42	3957.0	4419.69	4169.29	23 min 42 sec

Tabulka 9. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point penalization 0	4869.56	4604.0	4893.39	4827.0	20 min 53 sec
KS cross 2 point penalization 0.1	4835.53	4217.0	4870.1	4799.59	18 min 39 sec
KS cross 2 point penalization 0.2	4789.75	4703.0	4850.8	4705.49	19 min 1 sec
KS cross 2 point penalization 0.3	4746.55	4555.0	4877.8	4608.89	20 min 2 sec
KS cross 2 point penalization 0.4	4768.5	4239.0	4808.0	4706.39	20 min 24 sec
KS cross 2 point penalization 0.5	4697.37	4287.0	4771.2	4578.29	21 min 31 sec
KS cross 2 point penalization 0.6	4662.39	4761.0	4760.89	4577.89	21 min 17 sec
KS cross 2 point penalization 0.7	4597.61	4489.0	4674.59	4491.5	22 min 44 sec
KS cross 2 point penalization 0.8	4511.39	4365.0	4580.39	4391.59	24 min 5 sec
KS cross 2 point penalization 0.9	4465.93	4377.0	4545.2	4410.09	24 min 39 sec
KS cross 2 point penalization 1	4363.87	3631.0	4533.3	4138.8	24 min 9 sec

Tabulka 10. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 pen	4857.01	4294.0	4894.39	4838.7	20 min 24 sec
KS Uniform crossover 0.1 pen	4849.41	3831.0	4897.7	4794.7	16 min 33 sec
KS Uniform crossover 0.2 pen	4823.45	4305.0	4865.29	4784.09	17 min 0 sec
KS Uniform crossover 0.3 pen	4847.89	4647.0	4868.89	4809.3	17 min 55 sec
KS Uniform crossover 0.4 pen	4824.39	3903.0	4868.2	4751.49	17 min 50 sec
KS Uniform crossover 0.5 pen	4860.78	4623.0	4897.5	4815.4	18 min 27 sec
KS Uniform crossover 0.6 pen	4839.95	4451.0	4875.59	4789.59	20 min 4 sec
KS Uniform crossover 0.7 pen	4843.05	4321.0	4892.8	4789.1	22 min 26 sec
KS Uniform crossover 0.8 pen	4836.55	3231.0	4879.59	4811.79	22 min 16 sec
KS Uniform crossover 0.9 pen	4820.39	4393.0	4844.4	4778.39	24 min 53 sec
KS Uniform crossover 1 pen	4869.74	4159.0	4899.5	4829.8	23 min 57 sec

Tabulka 11. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut 0 corr	4959.84	43.0	4963.6	4957.2	19 min 1 sec
KS mut 0.01 corr	4963.68	2197.0	4964.0	4963.6	23 min 31 sec
KS mut 0.02 corr	4960.42	2439.0	4963.6	4957.2	25 min 54 sec
KS mut 0.03 corr	4956.34	2705.0	4961.4	4945.1	28 min 52 sec
KS mut 0.04 corr	4952.72	3135.0	4961.4	4941.5	33 min 20 sec
KS mut 0.05 corr	4945.56	3359.0	4957.2	4934.5	34 min 42 sec
KS mut 0.06 corr	4938.18	3055.0	4948.5	4928.0	36 min 53 sec
KS mut 0.07 corr	4931.78	3931.0	4957.5	4912.6	39 min 14 sec
KS mut 0.08 corr	4906.28	3961.0	4927.5	4881.7	43 min 18 sec
KS mut 0.09 corr	4914.62	3979.0	4925.0	4904.8	42 min 50 sec
KS mut 0.1 corr	4872.18	3849.0	4923.7	4850.0	43 min 6 sec
KS mut 0.2 corr	4632.66	3513.0	4699.89	4582.4	78 min 39 sec
KS mut 0.4 corr	4438.89	3051.0	4531.89	4363.09	43 min 28 sec
KS mut 0.8 corr	4380.37	3087.0	4414.39	4352.89	40 min 5 sec
KS mut 1 corr	4360.39	3623.0	4409.0	4316.1	25 min 0 sec

Tabulka 12. Test 4, pokračování

Název souboru:	cross_mut_pack.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut penal 0	4117.92	553.0	4312.29	3834.39	21 min 8 sec
KS mut penal 0.01	4460.13	3691.0	4608.7	4297.79	25 min 44 sec
KS mut penal 0.02	4492.85	4501.0	4555.89	4418.09	24 min 43 sec
KS mut penal 0.03	4216.51	4733.0	4428.09	4106.1	21 min 57 sec
KS mut penal 0.04	4113.12	4209.0	4283.69	3872.89	20 min 47 sec
KS mut penal 0.05	3847.82	4187.0	3956.19	3714.79	30 min 4 sec
KS mut penal 0.06	3611.69	4245.0	3923.59	3470.99	31 min 30 sec
KS mut penal 0.07	3557.03	4707.0	3904.19	3314.59	20 min 18 sec
KS mut penal 0.08	3282.63	3897.0	3631.09	3078.3	22 min 41 sec
KS mut penal 0.09	2973.01	3633.0	3072.49	2792.69	22 min 41 sec
KS mut penal 0.1	2662.33	3789.0	2873.8	2477.59	22 min 23 sec
KS mut penal 0.2	-80252.95	3797.0	0.0	-105098.19	23 min 26 sec
KS mut penal 0.4	-150533.26	2951.0	0.0	-161022.8	25 min 35 sec
KS mut penal 0.8	-161019.76	2305.0	0.0	-177496.3	23 min 19 sec
KS mut penal 1	-173171.8	3451.0	0.0	-184568.7	23 min 31 sec

Tabulka 13. Test 4, pokračování

Zjistili jsme, že použití korekce nepřípustných řetězců vede ke zhruba stejným výsledkům pro všechny způsoby křížení. Výrazněji se projevila pouze mutace, která se jeví jako vhodná v rozmezí 1 až 2 procent. Zajímavěji dopadla varianta fitness s penalizací nepřípustných řetězců. Vzhledem k ohromnému počtu nepřípustných řetězců v generaci (na počátku téměř všechny), došlo k tomu, že křížení v jednom i dvou bodech vedlo ke zhoršení nalezeného výsledku. Pouze destruktivnější uniformní křížení dopadlo o něco lépe. Největší roli ale v tomto výsledku hraje způsob selekce a tím byla vážená ruleta, která si evidentně dobře s velkým rozmězím hodnot fitness v tak malém počtu generací neporadila. Proto test zopakujeme použitím selekce turnajem.

Test 5
Nyní zopakujeme test 4 s použitím selekce turnajem.

prst. křížení	prst. mutace	Elitářů	Kroků	Populace	Selekce	Křížení
testována	0.01	1	5000	500	Turnaj 7	testováno

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point 0 corr	4963.16	2849.0	4963.6	4962.3	13 min 41 sec
KS cross 1 point 0.1 corr	4962.82	1903.0	4963.6	4962.3	11 min 22 sec
KS cross 1 point 0.2 corr	4963.08	2089.0	4963.6	4962.3	12 min 1 sec
KS cross 1 point 0.3 corr	4963.16	643.0	4964.0	4962.3	12 min 29 sec
KS cross 1 point 0.4 corr	4963.16	2363.0	4963.6	4961.4	12 min 41 sec
KS cross 1 point 0.5 corr	4963.34	1887.0	4963.6	4962.3	13 min 2 sec
KS cross 1 point 0.6 corr	4962.82	2223.0	4963.6	4962.3	13 min 23 sec
KS cross 1 point 0.7 corr	4962.64	1933.0	4963.6	4961.4	13 min 43 sec
KS cross 1 point 0.8 corr	4963.08	1359.0	4963.6	4962.3	14 min 6 sec
KS cross 1 point 0.9 corr	4962.9	2831.0	4963.6	4961.4	14 min 21 sec
KS cross 1 point 1 corr	4963.08	2349.0	4963.6	4962.3	14 min 44 sec

Tabulka 14. Test 5

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point 0 corr	4963.16	1937.0	4963.6	4962.3	13 min 44 sec
KS cross 2 point 0.1 corr	4962.28	3201.0	4964.0	4961.4	11 min 46 sec
KS cross 2 point 0.2 corr	4962.9	1983.0	4964.0	4962.3	12 min 11 sec
KS cross 2 point 0.3 corr	4962.56	875.0	4963.6	4962.3	12 min 31 sec
KS cross 2 point 0.4 corr	4963.08	1333.0	4963.6	4962.3	12 min 44 sec
KS cross 2 point 0.5 corr	4962.9	1427.0	4964.0	4962.3	13 min 14 sec
KS cross 2 point 0.6 corr	4962.56	1043.0	4963.6	4962.3	13 min 26 sec
KS cross 2 point 0.7 corr	4962.56	2711.0	4963.6	4962.3	13 min 46 sec
KS cross 2 point 0.8 corr	4963.42	677.0	4964.0	4962.3	14 min 14 sec
KS cross 2 point 0.9 corr	4963.08	1005.0	4963.6	4962.3	14 min 26 sec
KS cross 2 point 1 corr	4963.6	3079.0	4963.6	4963.6	14 min 49 sec

Tabulka 15. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 corr	4963.73	2949.0	4964.0	4963.6	13 min 56 sec
KS Uniform crossover 0.1 corr	4963.16	1045.0	4964.0	4962.3	11 min 49 sec
KS Uniform crossover 0.2 corr	4963.42	1139.0	4964.0	4962.3	12 min 10 sec
KS Uniform crossover 0.3 corr	4963.34	1673.0	4963.6	4962.3	12 min 22 sec
KS Uniform crossover 0.4 corr	4963.42	2071.0	4964.0	4962.3	12 min 58 sec
KS Uniform crossover 0.5 corr	4963.16	2907.0	4964.0	4962.3	13 min 13 sec
KS Uniform crossover 0.6 corr	4963.34	1507.0	4963.6	4962.3	13 min 22 sec
KS Uniform crossover 0.7 corr	4963.76	909.0	4964.0	4963.6	13 min 57 sec
KS Uniform crossover 0.8 corr	4963.42	839.0	4964.0	4962.3	14 min 11 sec
KS Uniform crossover 0.9 corr	4962.64	2389.0	4963.6	4961.4	14 min 22 sec
KS Uniform crossover 1 corr	4962.9	1803.0	4964.0	4962.3	14 min 51 sec

Tabulka 16. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point penalization 0	4959.6	2581.0	4963.6	4955.9	13 min 9 sec
KS cross 1 point penalization 0.1	4960.08	3155.0	4963.6	4953.2	11 min 20 sec
KS cross 1 point penalization 0.2	4961.86	2159.0	4964.0	4957.9	11 min 37 sec
KS cross 1 point penalization 0.3	4961.88	2529.0	4962.3	4960.2	11 min 56 sec
KS cross 1 point penalization 0.4	4962.6	2465.0	4964.0	4960.4	12 min 18 sec
KS cross 1 point penalization 0.5	4961.66	1661.0	4963.6	4959.7	12 min 40 sec
KS cross 1 point penalization 0.6	4963.24	1439.0	4964.0	4962.3	12 min 58 sec
KS cross 1 point penalization 0.7	4962.28	945.0	4964.0	4960.4	13 min 20 sec
KS cross 1 point penalization 0.8	4962.8	1131.0	4964.0	4962.3	13 min 40 sec
KS cross 1 point penalization 0.9	4962.89	3237.0	4964.0	4962.3	14 min 0 sec
KS cross 1 point penalization 1	4962.64	1491.0	4964.0	4962.3	14 min 20 sec

Tabulka 17. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point penalization 0	4960.58	2241.0	4962.3	4957.8	13 min 12 sec
KS cross 2 point penalization 0.1	4962.85	2693.0	4964.0	4962.1	11 min 32 sec
KS cross 2 point penalization 0.2	4962.51	2551.0	4964.0	4960.4	11 min 40 sec
KS cross 2 point penalization 0.3	4962.4	2315.0	4963.6	4961.1	12 min 0 sec
KS cross 2 point penalization 0.4	4962.01	1887.0	4962.3	4961.1	12 min 20 sec
KS cross 2 point penalization 0.5	4962.89	2897.0	4964.0	4962.3	12 min 40 sec
KS cross 2 point penalization 0.6	4962.66	1425.0	4964.0	4961.1	13 min 0 sec
KS cross 2 point penalization 0.7	4962.44	2147.0	4964.0	4959.8	13 min 20 sec
KS cross 2 point penalization 0.8	4961.76	2579.0	4964.0	4957.9	13 min 40 sec
KS cross 2 point penalization 0.9	4962.18	1045.0	4963.6	4960.4	14 min 0 sec
KS cross 2 point penalization 1	4962.44	799.0	4963.6	4960.4	14 min 20 sec

Tabulka 18. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 pen	4962.51	2361.0	4964.0	4960.4	13 min 13 sec
KS Uniform crossover 0.1 pen	4961.62	2025.0	4962.3	4960.4	11 min 7 sec
KS Uniform crossover 0.2 pen	4961.28	2301.0	4963.6	4955.8	11 min 26 sec
KS Uniform crossover 0.3 pen	4962.26	3373.0	4964.0	4961.1	12 min 4 sec
KS Uniform crossover 0.4 pen	4960.24	3203.0	4962.3	4957.9	12 min 26 sec
KS Uniform crossover 0.5 pen	4959.7	2371.0	4961.1	4957.9	12 min 30 sec
KS Uniform crossover 0.6 pen	4959.87	1969.0	4962.3	4956.3	13 min 6 sec
KS Uniform crossover 0.7 pen	4961.12	3045.0	4963.6	4958.8	13 min 11 sec
KS Uniform crossover 0.8 pen	4961.87	3227.0	4962.3	4960.2	13 min 44 sec
KS Uniform crossover 0.9 pen	4961.24	3191.0	4962.3	4960.1	14 min 4 sec
KS Uniform crossover 1 pen	4961.87	1427.0	4964.0	4957.19	14 min 8 sec

Tabulka 19. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut 0 corr	4918.34	11.0	4952.8	4886.9	11 min 22 sec
KS mut 0.01 corr	4962.38	1627.0	4963.6	4961.4	13 min 55 sec
KS mut 0.02 corr	4963.76	2799.0	4964.0	4963.6	16 min 13 sec
KS mut 0.03 corr	4963.92	2003.0	4964.0	4963.6	18 min 18 sec
KS mut 0.04 corr	4963.6	511.0	4963.6	4963.6	20 min 24 sec
KS mut 0.05 corr	4962.02	2199.0	4963.6	4961.4	22 min 31 sec
KS mut 0.06 corr	4959.44	1737.0	4962.1	4956.2	24 min 40 sec
KS mut 0.07 corr	4958.16	2929.0	4960.2	4955.4	26 min 48 sec
KS mut 0.08 corr	4954.39	3257.0	4961.4	4944.3	28 min 58 sec
KS mut 0.09 corr	4942.38	3879.0	4958.6	4921.8	30 min 45 sec
KS mut 0.1 corr	4936.3	2843.0	4954.8	4922.5	32 min 45 sec
KS mut 0.2 corr	4743.36	3103.0	4776.7	4726.3	54 min 19 sec
KS mut 0.4 corr	4462.42	1473.0	4493.5	4437.19	99 min 21 sec
KS mut 0.8 corr	4278.42	3897.0	4284.2	4269.49	193 min 49 sec
KS mut 1 corr	4073.57	0.0	4141.0	4038.59	241 min 45 sec

Tabulka 20. Test 5, pokračování

Název souboru:	cross_mut_pack_tour7.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut penal 0	3799.43	23.0	4149.79	3316.8	13 min 8 sec
KS mut penal 0.01	4962.01	2117.0	4964.0	4957.9	13 min 5 sec
KS mut penal 0.02	4884.76	4247.0	4918.3	4852.39	13 min 5 sec
KS mut penal 0.03	4728.18	4457.0	4764.0	4703.2	13 min 5 sec
KS mut penal 0.04	4496.76	4519.0	4575.2	4429.09	13 min 5 sec
KS mut penal 0.05	4324.15	4081.0	4499.29	4186.99	13 min 7 sec
KS mut penal 0.06	4073.69	3875.0	4147.09	3984.79	13 min 10 sec
KS mut penal 0.07	3870.12	4161.0	3965.6	3769.69	13 min 10 sec
KS mut penal 0.08	3723.89	4195.0	3986.19	3406.39	13 min 15 sec
KS mut penal 0.09	3606.99	4339.0	3770.59	3517.79	13 min 15 sec
KS mut penal 0.1	3359.47	4193.0	3541.39	3266.1	13 min 15 sec
KS mut penal 0.2	-56333.86	2913.0	0.0	-66609.99	13 min 30 sec
KS mut penal 0.4	-140831.84	4027.0	0.0	-153606.8	13 min 50 sec
KS mut penal 0.8	-182395.99	3033.0	0.0	-205072.89	13 min 50 sec
KS mut penal 1	-248867.94	0.0	0.0	-258845.3	13 min 44 sec

Tabulka 21. Test 5, pokračování

V tomto testu se projevil význam křížení ve variantě s penalizací, nejlépe dopadlo křížení v jednom bodě s pravděpodobností okolo 0.5 - 0.9, poté křížení ve dvou bodech s pravděpodobností okolo 0.5 a nejhůře pak uniformní křížení.

Test 6
Zopakujeme předchozí test ještě jednou se stejnými daty, tentokráte ale pro populaci o velikosti 100.

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point 0 corr	4961.48	3304.0	4963.6	4960.1	4 min 51 sec
KS cross 1 point 0.1 corr	4959.78	2757.0	4964.0	4950.2	4 min 3 sec
KS cross 1 point 0.2 corr	4961.76	4659.0	4963.6	4960.1	3 min 30 sec
KS cross 1 point 0.3 corr	4961.24	3947.0	4963.6	4957.8	3 min 54 sec
KS cross 1 point 0.4 corr	4961.18	1471.0	4964.0	4956.2	4 min 1 sec
KS cross 1 point 0.5 corr	4961.88	4117.0	4963.6	4959.8	4 min 1 sec
KS cross 1 point 0.6 corr	4958.6	3003.0	4963.6	4950.2	4 min 10 sec
KS cross 1 point 0.7 corr	4963.06	1769.0	4964.0	4961.4	4 min 15 sec
KS cross 1 point 0.8 corr	4962.48	3433.0	4964.0	4961.1	4 min 22 sec
KS cross 1 point 0.9 corr	4962.14	2435.0	4963.6	4961.1	4 min 37 sec
KS cross 1 point 1 corr	4962.14	2925.0	4963.6	4959.8	4 min 33 sec

Tabulka 22. Test 6

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point 0 corr	4962.13	2942.0	4964.0	4961.4	4 min 21 sec
KS cross 2 point 0.1 corr	4963.24	3975.0	4964.0	4961.4	3 min 42 sec
KS cross 2 point 0.2 corr	4962.8	4225.0	4964.0	4961.4	3 min 49 sec
KS cross 2 point 0.3 corr	4962.64	4241.0	4963.6	4961.4	3 min 54 sec
KS cross 2 point 0.4 corr	4962.88	2461.0	4964.0	4961.4	4 min 1 sec
KS cross 2 point 0.5 corr	4962.0	4595.0	4963.6	4960.4	4 min 35 sec
KS cross 2 point 0.6 corr	4962.12	2353.0	4962.3	4961.4	4 min 41 sec
KS cross 2 point 0.7 corr	4962.38	3537.0	4963.6	4961.4	4 min 47 sec
KS cross 2 point 0.8 corr	4962.1	2491.0	4964.0	4961.4	4 min 34 sec
KS cross 2 point 0.9 corr	4961.4	4511.0	4961.4	4961.4	4 min 30 sec
KS cross 2 point 1 corr	4962.18	2169.0	4963.6	4960.4	4 min 45 sec

Tabulka 23. Test 6, pokračování

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 corr	4962.65	3609.0	4964.0	4961.4	4 min 17 sec
KS Uniform crossover 0.1 corr	4961.96	2697.0	4964.0	4960.1	3 min 44 sec
KS Uniform crossover 0.2 corr	4962.02	2783.0	4963.6	4961.4	3 min 46 sec
KS Uniform crossover 0.3 corr	4962.04	2953.0	4964.0	4959.8	3 min 58 sec
KS Uniform crossover 0.4 corr	4960.8	3629.0	4962.3	4957.5	4 min 14 sec
KS Uniform crossover 0.5 corr	4961.34	3739.0	4962.3	4960.2	4 min 31 sec
KS Uniform crossover 0.6 corr	4961.74	4707.0	4963.6	4957.8	4 min 32 sec
KS Uniform crossover 0.7 corr	4961.96	2821.0	4963.6	4960.1	4 min 26 sec
KS Uniform crossover 0.8 corr	4961.58	2041.0	4963.6	4960.1	4 min 26 sec
KS Uniform crossover 0.9 corr	4961.0	4769.0	4963.6	4957.8	4 min 34 sec
KS Uniform crossover 1	4962.14	3843.0	4963.6	4959.8	4 min 35 sec

Tabulka 24. Test 6, pokračování

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 1 point penalization 0	4953.8	3434.0	4964.0	4943.9	4 min 11 sec
KS cross 1 point penalization 0.1	4953.46	3169.0	4959.8	4946.6	3 min 36 sec
KS cross 1 point penalization 0.2	4954.88	3355.0	4961.1	4946.39	3 min 42 sec
KS cross 1 point penalization 0.3	4953.44	3055.0	4960.9	4946.7	4 min 22 sec
KS cross 1 point penalization 0.4	4954.7	3653.0	4961.4	4943.5	4 min 1 sec
KS cross 1 point penalization 0.5	4962.12	3337.0	4964.0	4960.8	3 min 55 sec
KS cross 1 point penalization 0.6	4955.26	3303.0	4962.3	4944.3	4 min 1 sec
KS cross 1 point penalization 0.7	4956.68	2043.0	4962.7	4951.3	4 min 9 sec
KS cross 1 point penalization 0.8	4961.18	2589.0	4963.6	4959.8	4 min 24 sec
KS cross 1 point penalization 0.9	4960.08	2971.0	4962.3	4957.9	4 min 21 sec
KS cross 1 point penalization 1	4958.1	2821.0	4962.3	4952.1	4 min 48 sec

Tabulka 25. Test 6, pokračování

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point penalization 0	4955.1	3519.0	4962.7	4949.7	4 min 6 sec
KS cross 2 point penalization 0.1	4951.34	3599.0	4960.8	4936.1	3 min 30 sec
KS cross 2 point penalization 0.2	4954.88	3005.0	4959.9	4951.5	3 min 38 sec
KS cross 2 point penalization 0.3	4956.18	2567.0	4961.4	4950.4	3 min 46 sec
KS cross 2 point penalization 0.4	4953.3	2557.0	4960.8	4946.1	3 min 51 sec
KS cross 2 point penalization 0.5	4960.48	2445.0	4964.0	4951.3	3 min 59 sec
KS cross 2 point penalization 0.6	4959.54	1239.0	4960.8	4955.1	4 min 9 sec
KS cross 2 point penalization 0.7	4958.5	2325.0	4964.0	4954.2	4 min 10 sec
KS cross 2 point penalization 0.8	4957.4	2671.0	4962.3	4953.1	4 min 23 sec
KS cross 2 point penalization 0.9	4956.72	2375.0	4962.7	4949.89	4 min 31 sec
KS cross 2 point penalization 1	4958.97	1879.0	4962.7	4956.0	4 min 34 sec

Tabulka 26. Test 6, pokračování

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 pen	4957.51	3492.0	4962.1	4947.3	3 min 39 sec
KS Uniform crossover 0.1 pen	4953.18	4175.0	4962.3	4945.39	3 min 2 sec
KS Uniform crossover 0.2 pen	4953.42	3005.0	4961.4	4943.9	3 min 11 sec
KS Uniform crossover 0.3 pen	4952.16	3647.0	4957.2	4949.0	3 min 14 sec
KS Uniform crossover 0.4 pen	4955.24	2109.0	4962.1	4944.2	3 min 17 sec
KS Uniform crossover 0.5 pen	4949.26	3703.0	4956.2	4936.6	3 min 23 sec
KS Uniform crossover 0.6 pen	4956.74	3465.0	4963.6	4947.4	3 min 27 sec
KS Uniform crossover 0.7 pen	4958.16	3561.0	4962.7	4953.5	3 min 30 sec
KS Uniform crossover 0.8 pen	4956.7	3191.0	4960.8	4949.3	3 min 43 sec
KS Uniform crossover 0.9 pen	4954.28	2509.0	4959.8	4947.6	3 min 40 sec
KS Uniform crossover 1 pen	4953.92	3779.0	4962.1	4944.0	3 min 48 sec

Tabulka 27. Test 6, pokračování

Název souboru	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut 0 corr	4623.8	5.0	4772.1	4558.49	3 min 8 sec
KS mut 0.01 corr	4962.72	2745.0	4964.0	4961.4	3 min 48 sec
KS mut 0.02 corr	4963.68	2789.0	4964.0	4963.6	4 min 42 sec
KS mut 0.03 corr	4963.6	317.0	4963.6	4963.6	5 min 38 sec
KS mut 0.04 corr	4963.34	1843.0	4963.6	4962.3	5 min 49 sec
KS mut 0.05 corr	4962.4	2479.0	4963.6	4961.1	6 min 28 sec
KS mut 0.06 corr	4959.0	2593.0	4961.4	4952.5	6 min 50 sec
KS mut 0.07 corr	4960.5	2899.0	4961.4	4957.5	8 min 1 sec
KS mut 0.08 corr	4953.84	2839.0	4961.4	4930.6	8 min 14 sec
KS mut 0.09 corr	4936.76	3947.0	4951.4	4923.8	8 min 50 sec
KS mut 0.1 corr	4937.34	3071.0	4960.0	4910.0	9 min 28 sec
KS mut 0.2 corr	4713.47	4043.0	4739.9	4677.79	14 min 39 sec
KS mut 0.4 corr	4449.29	2559.0	4491.8	4416.7	26 min 1 sec
KS mut 0.8 corr	4214.62	2487.0	4276.89	4168.69	51 min 29 sec
KS mut 1 corr	4002.5	659.0	4042.2	3951.6	62 min 39 sec

Tabulka 28. Test 6, pokračování

Název souboru:	cross_mut_pack_tour7_pop100.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut penal 0	-61381.32	9.0	0.0	-122670.8	4 min 31 sec
KS mut penal 0.01	4957.36	3863.0	4962.3	4953.7	5 min 14 sec
KS mut penal 0.02	4890.52	4525.0	4916.2	4836.2	4 min 8 sec
KS mut penal 0.03	4707.05	4565.0	4769.4	4665.39	4 min 6 sec
KS mut penal 0.04	4515.92	3879.0	4604.89	4457.7	4 min 10 sec
KS mut penal 0.05	4296.97	4251.0	4356.09	4242.89	4 min 16 sec
KS mut penal 0.06	4118.41	3853.0	4259.09	3901.79	4 min 5 sec
KS mut penal 0.07	3895.27	3773.0	4006.39	3802.59	4 min 6 sec
KS mut penal 0.08	3662.65	4007.0	3918.79	3449.89	4 min 4 sec
KS mut penal 0.09	3605.09	3831.0	3687.79	3501.99	4 min 4 sec
KS mut penal 0.1	3324.61	4181.0	3590.39	3108.99	4 min 19 sec
KS mut penal 0.2	-50935.44	3069.0	0.0	-77651.1	4 min 30 sec
KS mut penal 0.4	-153761.74	3341.0	0.0	-171976.3	4 min 27 sec
KS mut penal 0.8	-202286.21	2155.0	0.0	-210479.19	4 min 39 sec
KS mut penal 1	-268269.8	0.0	0.0	-288435.5	4 min 34 sec

Tabulka 29. Test 6, pokračování

Při snížené velikosti populace se také snížily nalezené hodnoty. U varianty s penalizací se zdá výhodné křížení s pravděpodobností 0.5 - 0.9. U varianty s korekcí se opět neukázala žádná varianta křížení jako nejlepší.

Test 7

Nyní provedeme opět stejný test, tentokráte ovšem na jiných datech. Variantu s korekcí jsem vynechal, protože výsledky byly téměř rovnocenné pro všechna nastavení.

Název souboru:	cross_mut_pack_tour7_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepší	kroků	nejlepší	nejhorší	čas
KS cross 1 point penalization 0	5576.28	4187.0	5594.19	5557.69	15 min 54 sec
KS cross 1 point penalization 0.1	5584.23	2951.0	5595.09	5575.59	13 min 40 sec
KS cross 1 point penalization 0.2	5583.63	3839.0	5602.29	5572.59	14 min 5 sec
KS cross 1 point penalization 0.3	5587.33	3455.0	5595.29	5576.69	14 min 30 sec
KS cross 1 point penalization 0.4	5582.53	3837.0	5587.79	5576.49	14 min 55 sec
KS cross 1 point penalization 0.5	5586.71	3473.0	5596.19	5576.49	15 min 21 sec
KS cross 1 point penalization 0.6	5592.11	3219.0	5596.59	5585.19	15 min 56 sec
KS cross 1 point penalization 0.7	5583.99	3743.0	5589.69	5570.09	16 min 14 sec
KS cross 1 point penalization 0.8	5585.47	2931.0	5594.79	5571.39	16 min 40 sec
KS cross 1 point penalization 0.9	5589.19	3179.0	5596.59	5579.09	17 min 5 sec
KS cross 1 point penalization 1	5585.19	2747.0	5593.79	5575.99	17 min 30 sec

Tabulka 30. Test 7, pokračování

Název souboru:	cross_mut_pack_tour7_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS cross 2 point penalization 0	5575.01	4016.0	5590.19	5565.39	15 min 54 sec
KS cross 2 point penalization 0.1	5580.39	3535.0	5588.89	5572.49	13 min 40 sec
KS cross 2 point penalization 0.2	5588.71	4225.0	5603.79	5582.09	14 min 5 sec
KS cross 2 point penalization 0.3	5590.73	3273.0	5598.59	5581.29	14 min 30 sec
KS cross 2 point penalization 0.4	5588.39	2995.0	5603.79	5581.49	14 min 58 sec
KS cross 2 point penalization 0.5	5586.23	2737.0	5596.59	5579.49	15 min 25 sec
KS cross 2 point penalization 0.6	5587.65	3245.0	5593.49	5579.99	15 min 50 sec
KS cross 2 point penalization 0.7	5584.49	3119.0	5595.49	5577.69	16 min 15 sec
KS cross 2 point penalization 0.8	5585.99	3751.0	5591.99	5577.19	16 min 41 sec
KS cross 2 point penalization 0.9	5593.71	2531.0	5603.79	5585.89	17 min 8 sec
KS cross 2 point penalization 1	5592.63	2505.0	5598.19	5582.29	17 min 34 sec

Tabulka 31. Test 7, pokračování

Název souboru:	cross_mut_pack_tour7_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS Uniform crossover 0 pen	5582.16	4037.0	5603.79	5567.89	15 min 46 sec
KS Uniform crossover 0.1 pen	5585.19	3739.0	5588.79	5576.69	13 min 33 sec
KS Uniform crossover 0.2 pen	5573.79	3213.0	5585.79	5569.89	14 min 0 sec
KS Uniform crossover 0.3 pen	5582.35	3667.0	5593.89	5562.79	14 min 32 sec
KS Uniform crossover 0.4 pen	5575.99	2889.0	5594.39	5547.39	14 min 55 sec
KS Uniform crossover 0.5 pen	5580.09	3757.0	5588.89	5574.79	15 min 24 sec
KS Uniform crossover 0.6 pen	5583.45	3669.0	5589.69	5575.09	15 min 50 sec
KS Uniform crossover 0.7 pen	5581.71	2199.0	5595.19	5568.59	16 min 15 sec
KS Uniform crossover 0.8 pen	5589.15	4141.0	5596.59	5578.59	17 min 28 sec
KS Uniform crossover 0.9 pen	5575.95	3769.0	5585.69	5566.39	17 min 15 sec
KS Uniform crossover 1 pen	5581.47	3835.0	5586.69	5571.59	17 min 34 sec

Tabulka 32. Test 7, pokračování

Název souboru:	cross_mut_pack_tour7_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut 0 corr	5554.91	11.0	5584.89	5513.59	13 min 53 sec
KS mut 0.01 corr	5603.79	1393.0	5603.79	5603.79	17 min 15 sec
KS mut 0.02 corr	5603.79	4927.0	5603.79	5603.79	20 min 25 sec
KS mut 0.03 corr	5603.79	1067.0	5603.79	5603.79	23 min 10 sec
KS mut 0.04 corr	5602.33	793.0	5603.79	5596.49	26 min 1 sec
KS mut 0.05 corr	5598.37	2683.0	5603.79	5593.19	29 min 4 sec
KS mut 0.06 corr	5589.59	2781.0	5592.59	5585.39	31 min 50 sec
KS mut 0.07 corr	5589.41	3835.0	5602.09	5578.19	34 min 42 sec
KS mut 0.08 corr	5556.85	3991.0	5577.49	5533.09	37 min 56 sec
KS mut 0.09 corr	5544.87	3225.0	5565.59	5503.49	40 min 53 sec
KS mut 0.1 corr	5511.51	3323.0	5547.79	5457.19	44 min 13 sec
KS mut 0.2 corr	5229.43	3303.0	5281.49	5179.49	73 min 2 sec
KS mut 0.4 corr	4976.07	2187.0	5006.29	4942.89	135 min 51 sec
KS mut 0.8 corr	4774.39	2821.0	4848.99	4742.09	271 min 39 sec
KS mut 1 corr	4510.08	5.0	4583.79	4482.39	63 min 41 sec

Tabulka 33. Test 7, pokračování

Název souboru:	cross_mut_pack_tour7_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	nejlepší	nejhorší	čas
KS mut penal 0	3743.82	25.0	3896.3	3590.4	16 min 55 sec
KS mut penal 0.01	5594.56	3645.0	5602.29	5581.79	16 min 59 sec
KS mut penal 0.02	5406.55	4341.0	5463.39	5325.29	16 min 35 sec
KS mut penal 0.03	5054.45	4871.0	5215.19	4965.49	16 min 30 sec
KS mut penal 0.04	4690.3	4291.0	4966.79	4548.89	16 min 39 sec
KS mut penal 0.05	4424.18	4077.0	4486.89	4324.2	16 min 49 sec
KS mut penal 0.06	4216.82	4217.0	4309.03	4120.89	16 min 45 sec
KS mut penal 0.07	3757.2	3185.0	4087.9	3503.29	16 min 50 sec
KS mut penal 0.08	3708.4	3027.0	3971.4	3393.8	17 min 6 sec
KS mut penal 0.09	3244.62	3949.0	3545.7	2938.5	16 min 55 sec
KS mut penal 0.1	2354.61	4111.0	3314.2	-577.83	16 min 58 sec
KS mut penal 0.2	-47898.84	3099.0	0.0	-55034.16	17 min 30 sec
KS mut penal 0.4	-89779.22	2461.0	0.0	-96952.26	18 min 25 sec
KS mut penal 0.8	-123447.16	2595.0	0.0	-127454.23	18 min 18 sec
KS mut penal 1	-144522.56	-1.0	0.0	-151715.73	17 min 52 sec

Tabulka 34. Test 7, pokračování

V tomto testu se naprosto vytratil rozdíl mezi způsoby křížení u metody s korekcí. U metody s penalizací vyšlo nejlépe křížení ve dvou bodech s pravděpodobností 0.3, 0.9, 1.

Test 8

Poslední test bude zaměřen na kombinaci způsobů selekce a křížení. Pravděpodobnost křížení byla zvolena na 0.6. Mutace zůstává na 0.01, velikost populace bude 200 a počet kroků 5000. Pro tento test použijeme pouze penalizaci.

Název souboru:	ks_sel_cross_penal.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS simple 0.6 roulette	4675.68	4709.0	4765.8	4617.7	5 min 1 sec
KS simple 0.6 tour2	4805.55	3607.0	4853.8	4769.2	5 min 0 sec
KS simple 0.6 tour3	4913.35	3349.0	4925.6	4886.5	5 min 2 sec
KS simple 0.6 tour4	4940.77	3771.0	4951.5	4927.1	4 min 59 sec
KS simple 0.6 tour5	4946.42	3967.0	4954.1	4937.2	4 min 49 sec
KS simple 0.6 tour6	4961.22	2397.0	4964.0	4957.9	4 min 45 sec
KS simple 0.6 tour7	4959.99	2295.0	4962.7	4955.7	4 min 46 sec
KS simple 0.6 tour8	4961.78	3319.0	4964.0	4959.8	4 min 50 sec
KS simple 0.6 tour9	4961.76	2405.0	4963.6	4960.4	4 min 50 sec
KS simple 0.6 tour10	4956.52	2055.0	4960.8	4950.1	4 min 50 sec
KS simple 0.6 tour20	4956.52	3597.0	4962.3	4948.3	4 min 55 sec
KS simple 0.6 rank	4830.8	4699.0	4854.5	4787.59	5 min 2 sec
KS simple 0.6 random	4414.8	4439.0	4535.7	4260.19	4 min 45 sec

Tabulka 35. Test 8

Název souboru:	ks_sel_cross_penal.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS 2point 0.6 roulette	4684.32	4465.0	4724.1	4661.8	4 min 51 sec
KS 2point 0.6 tour2	4835.3	4233.0	4873.2	4802.39	4 min 45 sec
KS 2point 0.6 tour3	4912.82	3677.0	4923.89	4892.09	4 min 45 sec
KS 2point 0.6 tour4	4942.66	2601.0	4953.8	4935.3	4 min 45 sec
KS 2point 0.6 tour5	4958.54	3177.0	4960.8	4955.5	4 min 46 sec
KS 2point 0.6 tour6	4957.17	3485.0	4962.3	4947.2	4 min 45 sec
KS 2point 0.6 tour7	4960.18	2389.0	4963.6	4957.9	4 min 50 sec
KS 2point 0.6 tour8	4960.62	3121.0	4962.7	4957.9	4 min 50 sec
KS 2point 0.6 tour9	4960.98	1963.0	4962.7	4957.8	4 min 50 sec
KS 2point 0.6 tour10	4959.7	2775.0	4962.7	4955.7	4 min 50 sec
KS 2point 0.6 tour20	4952.55	2615.0	4960.8	4947.7	4 min 55 sec
KS 2point 0.6 rank	4823.13	4419.0	4838.2	4789.39	5 min 5 sec
KS 2point 0.6 random	4518.12	4727.0	4578.19	4457.39	4 min 45 sec

Tabulka 36. Test 8, pokračování

Název souboru:	ks_sel_cross_penal.test				
Předmětů:	200		Odhad:	4960.1	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS uniform 0.6 roulette	4838.68	4705.0	4883.9	4791.3	4 min 50 sec
KS uniform 0.6 tour2	4882.56	4063.0	4892.7	4870.7	4 min 45 sec
KS uniform 0.6 tour3	4914.76	3721.0	4931.2	4891.8	4 min 45 sec
KS uniform 0.6 tour4	4942.92	3499.0	4954.1	4936.6	4 min 45 sec
KS uniform 0.6 tour5	4952.52	3911.0	4960.6	4943.2	4 min 47 sec
KS uniform 0.6 tour6	4953.62	2545.0	4962.1	4948.1	4 min 50 sec
KS uniform 0.6 tour7	4957.56	2733.0	4961.4	4949.9	4 min 51 sec
KS uniform 0.6 tour8	4961.28	2711.0	4963.6	4960.2	4 min 50 sec
KS uniform 0.6 tour9	4959.05	2583.0	4963.6	4949.7	4 min 50 sec
KS uniform 0.6 tour10	4961.1	2281.0	4962.1	4960.4	4 min 50 sec
KS uniform 0.6 tour20	4959.08	2737.0	4962.3	4955.6	4 min 55 sec
KS uniform 0.6 rank	4876.85	4521.0	4926.5	4855.09	5 min 5 sec
KS uniform 0.6 random	4747.96	4485.0	4780.5	4713.09	4 min 45 sec

Tabulka 37. Test 8, pokračování

Nejlepších výsledků dosáhl výběr turnajem 7-10, o něco lépe s jednoduchým křížením, o něco hůře pak křížení ve dvou bodech a uniformní.

Test 9

Tento test je opakováním předchozího testu pro jiná data.

Název souboru:	ks_sel_cross_penal_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS simple 0.6 roulette	5195.93	4643.0	5372.89	5047.59	9 min 50 sec
KS simple 0.6 tour2	5310.13	4647.0	5418.29	5223.29	9 min 48 sec
KS simple 0.6 tour3	5458.45	4153.0	5496.89	5426.99	9 min 49 sec
KS simple 0.6 tour4	5508.51	4195.0	5540.39	5479.19	9 min 48 sec
KS simple 0.6 tour5	5556.43	4357.0	5592.29	5532.09	9 min 51 sec
KS simple 0.6 tour6	5570.15	3399.0	5588.39	5547.09	9 min 56 sec
KS simple 0.6 tour7	5581.65	3431.0	5586.99	5570.49	9 min 50 sec
KS simple 0.6 tour8	5586.57	2319.0	5593.29	5571.19	10 min 8 sec
KS simple 0.6 tour9	5586.93	3127.0	5590.49	5583.29	10 min 12 sec
KS simple 0.6 tour10	5596.19	2649.0	5602.29	5592.29	10 min 9 sec
KS simple 0.6 tour20	5594.83	2821.0	5597.99	5586.59	10 min 35 sec
KS simple 0.6 rank	5321.64	4447.0	5415.02	5251.99	10 min 5 sec
KS simple 0.6 random	4844.21	4523.0	4952.2	4718.89	9 min 44 sec

Tabulka 38. Test 8, pokračování

Název souboru:	ks_sel_cross_penal_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS 2point 0.6 roulette	5179.11	4485.0	5272.69	5128.49	9 min 50 sec
KS 2point 0.6 tour2	5305.43	4421.0	5352.59	5235.59	10 min 3 sec
KS 2point 0.6 tour3	5473.89	3543.0	5487.09	5457.99	10 min 11 sec
KS 2point 0.6 tour4	5510.07	3573.0	5543.79	5493.29	9 min 49 sec
KS 2point 0.6 tour5	5547.49	3197.0	5573.49	5526.09	9 min 46 sec
KS 2point 0.6 tour6	5565.39	3419.0	5580.79	5554.19	9 min 50 sec
KS 2point 0.6 tour7	5586.33	2635.0	5602.29	5569.59	10 min 19 sec
KS 2point 0.6 tour8	5586.29	3291.0	5597.99	5580.89	10 min 56 sec
KS 2point 0.6 tour9	5594.73	2151.0	5602.29	5587.89	10 min 34 sec
KS 2point 0.6 tour10	5589.75	1747.0	5602.29	5584.19	10 min 25 sec
KS 2point 0.6 tour20	5588.75	3435.0	5603.79	5577.69	10 min 25 sec
KS 2point 0.6 rank	5379.93	4277.0	5461.09	5346.69	11 min 44 sec
KS 2point 0.6 random	4762.01	4757.0	4864.19	4530.39	10 min 48 sec

Tabulka 39. Test 8, pokračování

Název souboru:	ks_sel_cross_penal_data2.test				
Předmětů:	250		Odhad:	5593.8	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
KS uniform 0.6 roulette	5384.87	4781.0	5443.59	5297.69	10 min 26 sec
KS uniform 0.6 tour2	5458.91	4471.0	5473.09	5439.69	12 min 39 sec
KS uniform 0.6 tour3	5493.49	4285.0	5549.19	5426.39	11 min 41 sec
KS uniform 0.6 tour4	5532.35	4159.0	5576.39	5504.79	11 min 47 sec
KS uniform 0.6 tour5	5550.01	4185.0	5587.59	5527.79	11 min 36 sec
KS uniform 0.6 tour6	5567.41	4359.0	5585.29	5555.59	11 min 48 sec
KS uniform 0.6 tour7	5565.85	4307.0	5578.89	5547.79	11 min 37 sec
KS uniform 0.6 tour8	5585.27	3689.0	5603.79	5577.59	11 min 40 sec
KS uniform 0.6 tour9	5584.05	3521.0	5595.49	5571.59	10 min 28 sec
KS uniform 0.6 tour10	5591.51	3621.0	5597.49	5584.99	10 min 15 sec
KS uniform 0.6 tour20	5596.29	1901.0	5603.79	5587.59	10 min 3 sec
KS uniform 0.6 rank	5437.66	4583.0	5466.29	5392.19	10 min 5 sec
KS uniform 0.6 random	5200.95	4695.0	5312.39	5074.59	9 min 46 sec

Tabulka 40. Test 8, pokračování

Stejně jako v předchozím testu nejhůře dopadlo křížení ve dvou bodech.

A.3 Testy SAT

SAT instance použité pro tyto testy vždy představují splnitelné formule. Tyto formule pocházejí z tzv. SATlib. Je možné je získat na adrese *http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html*. Hodnota fitness 1000 znamená nalezené řešení.

Test 1
V tomto testu se budeme zabývat selekcí. Použita je poměrně malá instance SAT problému. Ostatní parametry jsou: mutace 0.03, populace 500, kroků 5000, 2 point crossover 0.8.

Název souboru:	sat_small_sel.test				
Klauzulí	218		Proměnných	50	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
sat sel Roulette	1000.0	499.0	1000.0	1000.0	0 min 20 sec
sat sel tour 2	1000.0	345.0	1000.0	1000.0	0 min 10 sec
sat sel tour 3	998.16	145.0	1000.0	995.41	0 min 33 sec
sat sel tour 4	997.24	97.0	1000.0	995.41	0 min 52 sec
sat sel tour 5	999.08	195.0	1000.0	995.41	0 min 34 sec
sat sel tour 6	997.24	251.0	1000.0	995.41	0 min 56 sec
sat sel tour 7	999.08	161.0	1000.0	995.41	0 min 28 sec
sat sel tour 8	995.41	13.0	1000.0	990.82	1 min 9 sec
sat sel tour 9	997.24	191.0	1000.0	995.41	0 min 57 sec
sat sel tour 10	998.16	113.0	1000.0	995.41	0 min 36 sec
sat sel tour 20	998.16	41.0	1000.0	995.41	0 min 38 sec
sat sel rank	1000.0	241.0	1000.0	1000.0	0 min 22 sec
sat sel random	991.74	1049.0	995.41	986.23	1 min 14 sec

Tabulka 41. Test 1

Vidíme, že tato instance byla poměrně "jednoduchá" a pomocí všech metod, kromě náhodného výběru, bylo nalezeno řešení.

Test 2 Nyní test zopakujeme pro jiná data

Název souboru:	sat_small_sel_data2.test uf50-0657.cnf				
Klauzulí	218		Proměnných	50	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
sat sel Roulette	1000.0	433.0	1000.0	1000.0	0 min 19 sec
sat sel tour 2	1000.0	333.0	1000.0	1000.0	0 min 25 sec
sat sel tour 3	1000.0	103.0	1000.0	1000.0	0 min 6 sec
sat sel tour 4	1000.0	57.0	1000.0	1000.0	0 min 2 sec
sat sel tour 5	1000.0	203.0	1000.0	1000.0	0 min 14 sec
sat sel tour 6	1000.0	77.0	1000.0	1000.0	0 min 4 sec
sat sel tour 7	1000.0	23.0	1000.0	1000.0	0 min 0 sec
sat sel tour 8	999.08	61.0	1000.0	995.41	1 min 18 sec
sat sel tour 9	998.16	59.0	1000.0	995.41	2 min 41 sec
sat sel tour 10	998.16	45.0	1000.0	995.41	2 min 53 sec
sat sel tour 20	999.08	251.0	1000.0	995.41	2 min 6 sec
sat sel rank	1000.0	965.0	1000.0	1000.0	1 min 51 sec
sat sel random	997.24	2129.0	1000.0	995.41	6 min 25 sec

Tabulka 42. Test 2

Tato data také představují jednoduché zadání, opět bylo vyřešeno všemi metodami, kromě turnaje velikosti 8 a více a náhodného výběru.

Test 3
Naposledy zopakujeme test pro jiná data

Název souboru:	sat_small_sel_data3.test uf50-0657				
Klauzulí	218		Proměnných	50	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
sat sel Roulette	995.41	231.0	995.41	995.41	8 min 21 sec
sat sel tour 2	996.33	123.0	1000.0	995.41	5 min 26 sec
sat sel tour 3	995.41	19.0	995.41	995.41	6 min 47 sec
sat sel tour 4	996.33	71.0	1000.0	995.41	5 min 33 sec
sat sel tour 5	996.33	963.0	1000.0	995.41	6 min 13 sec
sat sel tour 6	995.41	187.0	1000.0	990.82	5 min 36 sec
sat sel tour 7	996.33	11.0	1000.0	995.41	5 min 42 sec
sat sel tour 8	997.24	469.0	1000.0	995.41	4 min 45 sec
sat sel tour 9	994.49	13.0	995.41	990.82	7 min 11 sec
sat sel tour 10	995.41	11.0	995.41	995.41	6 min 54 sec
sat sel tour 20	995.41	901.0	995.41	995.41	7 min 32 sec
sat sel rank	995.41	103.0	995.41	995.41	9 min 38 sec
sat sel random	993.57	1593.0	1000.0	986.23	6 min 33 sec

Tabulka 43. Test 3

Tato instance se zdá být poněkud těžší, přesto byla ve většině případů vyřešena. Nejlépe pak pomocí turnaje o velikosti 8.

Test 4

Test 4 je zaměřen na porovnání různých kombinací křížení a mutace. Pro selekci je použit tournament o velikosti 7. Algoritmus je spuštěn na 5000 generací s 500 jedinci v populaci.

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 cross 1 point 0	995.34	1134.0	997.67	993.02	13 min 19 sec
SAT mut 0.04 cross 1 point 0.1	997.67	965.0	1000.0	995.34	11 min 5 sec
SAT mut 0.04 cross 1 point 0.2	995.34	1169.0	997.67	988.37	11 min 40 sec
SAT mut 0.04 cross 1 point 0.3	997.2	2199.0	1000.0	995.34	13 min 10 sec
SAT mut 0.04 cross 1 point 0.4	996.27	1435.0	997.67	995.34	12 min 13 sec
SAT mut 0.04 cross 1 point 0.5	997.2	1477.0	997.67	995.34	12 min 26 sec
SAT mut 0.04 cross 1 point 0.6	997.2	2469.0	997.67	995.34	12 min 46 sec
SAT mut 0.04 cross 1 point 0.7	997.67	1273.0	1000.0	995.34	11 min 37 sec
SAT mut 0.04 cross 1 point 0.8	996.74	1503.0	997.67	995.34	14 min 15 sec
SAT mut 0.04 cross 1 point 0.9	996.74	925.0	1000.0	995.34	13 min 2 sec
SAT mut 0.04 cross 1 point 1	996.74	1593.0	1000.0	995.34	12 min 32 sec

Tabulka 44. Test 4

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 cross 2 point 0	996.12	1156.0	1000.0	993.02	11 min 29 sec
SAT mut 0.04 cross 2 point 0.1	996.74	1879.0	997.67	995.34	12 min 6 sec
SAT mut 0.04 cross 2 point 0.2	997.67	1499.0	1000.0	995.34	11 min 29 sec
SAT mut 0.04 cross 2 point 0.3	995.81	623.0	997.67	995.34	12 min 23 sec
SAT mut 0.04 cross 2 point 0.4	997.2	1329.0	1000.0	995.34	10 min 8 sec
SAT mut 0.04 cross 2 point 0.5	996.27	625.0	1000.0	993.02	10 min 53 sec
SAT mut 0.04 cross 2 point 0.6	996.27	573.0	997.67	995.34	13 min 5 sec
SAT mut 0.04 cross 2 point 0.7	996.74	1731.0	997.67	995.34	14 min 6 sec
SAT mut 0.04 cross 2 point 0.8	996.74	1647.0	997.67	995.34	13 min 48 sec
SAT mut 0.04 cross 2 point 0.9	997.2	1065.0	1000.0	995.34	13 min 39 sec
SAT mut 0.04 cross 2 point 1	997.2	963.0	1000.0	995.34	12 min 37 sec

Tabulka 45. Test 4, pokračování

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 Uniform cross 0	995.34	746.0	995.34	995.34	14 min 54 sec
SAT mut 0.04 Uniform cross 0.1	996.27	1573.0	997.67	995.34	12 min 11 sec
SAT mut 0.04 Uniform crossr 0.2	996.27	1699.0	997.67	995.34	12 min 35 sec
SAT mut 0.04 Uniform cross 0.3	997.2	1391.0	1000.0	995.34	12 min 31 sec
SAT mut 0.04 Uniform cross 0.4	996.74	957.0	997.67	995.34	11 min 21 sec
SAT mut 0.04 Uniform cross 0.5	996.74	1475.0	1000.0	995.34	11 min 49 sec
SAT mut 0.04 Uniform cross 0.6	997.2	1501.0	1000.0	995.34	12 min 8 sec
SAT mut 0.04 Uniform cross 0.7	996.74	1257.0	997.67	995.34	14 min 19 sec
SAT mut 0.04 Uniform cross 0.8	995.81	1409.0	997.67	993.02	14 min 8 sec
SAT mut 0.04 Uniform cross 0.9	998.13	3215.0	1000.0	995.34	11 min 44 sec
SAT mut 0.04 Uniform cross 1	996.27	1319.0	997.67	995.34	17 min 8 sec

Tabulka 46. Test 4, pokračování

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 cross 1 point 0	998.06	872.0	1000.0	995.34	10 min 8 sec
SAT mut 0.02 cross 1 point 0.1	995.81	85.0	997.67	995.34	12 min 53 sec
SAT mut 0.02 cross 1 point 0.2	997.67	345.0	997.67	997.67	12 min 44 sec
SAT mut 0.02 cross 1 point 0.3	996.27	273.0	997.67	995.34	11 min 14 sec
SAT mut 0.02 cross 1 point 0.4	997.67	1789.0	997.67	997.67	11 min 22 sec
SAT mut 0.02 cross 1 point 0.5	996.27	1101.0	997.67	995.34	11 min 49 sec
SAT mut 0.02 cross 1 point 0.6	996.74	535.0	997.67	993.02	14 min 5 sec
SAT mut 0.02 cross 1 point 0.7	995.81	67.0	997.67	995.34	13 min 5 sec
SAT mut 0.02 cross 1 point 0.8	996.74	705.0	997.67	995.34	14 min 27 sec
SAT mut 0.02 cross 1 point 0.9	996.27	295.0	997.67	995.34	14 min 3 sec
SAT mut 0.02 cross 1 point 1	996.27	113.0	1000.0	993.02	12 min 22 sec

Tabulka 47. Test 4, pokračování

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 cross 2 point 0	998.44	299.0	1000.0	997.67	9 min 46 sec
SAT mut 0.02 cross 2 point 0.1	997.67	1165.0	997.67	997.67	11 min 58 sec
SAT mut 0.02 cross 2 point 0.2	997.2	475.0	997.67	995.34	12 min 24 sec
SAT mut 0.02 cross 2 point 0.3	996.27	1155.0	997.67	993.02	12 min 19 sec
SAT mut 0.02 cross 2 point 0.4	997.2	1283.0	1000.0	995.34	10 min 21 sec
SAT mut 0.02 cross 2 point 0.5	995.34	995.0	995.34	995.34	12 min 58 sec
SAT mut 0.02 cross 2 point 0.6	995.34	111.0	997.67	993.02	13 min 5 sec
SAT mut 0.02 cross 2 point 0.7	996.74	659.0	997.67	995.34	13 min 20 sec
SAT mut 0.02 cross 2 point 0.8	995.34	77.0	995.34	995.34	13 min 38 sec
SAT mut 0.02 cross 2 point 0.9	996.74	721.0	1000.0	995.34	15 min 3 sec
SAT mut 0.02 cross 2 point 1	996.74	729.0	997.67	995.34	17 min 50 sec

Tabulka 48. Test 4, pokračování

Název souboru:	sat_cross_mut_tour7.test uf100_0528				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 Uniform cross 0	996.51	1174.0	997.67	995.34	15 min 13 sec
SAT mut 0.02 Uniform cross 0.1	996.27	507.0	997.67	995.34	12 min 35 sec
SAT mut 0.02 Uniform cross 0.2	997.67	1017.0	1000.0	995.34	10 min 32 sec
SAT mut 0.02 Uniform cross 0.3	997.67	805.0	1000.0	993.02	9 min 35 sec
SAT mut 0.02 Uniform cross 0.4	997.67	701.0	1000.0	995.34	10 min 27 sec
SAT mut 0.02 Uniform cross 0.5	996.74	659.0	997.67	995.34	15 min 32 sec
SAT mut 0.02 Uniform cross 0.6	997.2	1373.0	997.67	995.34	15 min 30 sec
SAT mut 0.02 Uniform cross 0.7	998.6	2243.0	1000.0	997.67	11 min 10 sec
SAT mut 0.02 Uniform cross 0.8	996.74	423.0	1000.0	993.02	13 min 47 sec
SAT mut 0.02 Uniform cross 0.9	997.67	1145.0	1000.0	995.34	13 min 15 sec
SAT mut 0.02 Uniform cross 1	998.13	555.0	1000.0	995.34	11 min 2 sec

Tabulka 49. Test 4, pokračování

Nejlépe dopadlo uniformní křížení s pravděpodobností přibližně 0.7 - 1.0, o něco lépe s pravděpodobností mutace 0.02 než 0.04

Test 5
Test 5 je opakování testu 4 pro jiná data.

Název souboru:	sat_cross_mut_tour7_2.test uf100-814				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 cross 1 point 0	995.73	1957.0	997.67	993.02	15 min 49 sec
SAT mut 0.04 cross 1 point 0.1	995.81	887.0	997.67	990.69	12 min 18 sec
SAT mut 0.04 cross 1 point 0.2	994.88	1395.0	997.67	993.02	12 min 22 sec
SAT mut 0.04 cross 1 point 0.3	996.27	1155.0	997.67	995.34	12 min 37 sec
SAT mut 0.04 cross 1 point 0.4	995.34	2383.0	997.67	993.02	13 min 5 sec
SAT mut 0.04 cross 1 point 0.5	996.74	655.0	997.67	995.34	14 min 14 sec
SAT mut 0.04 cross 1 point 0.6	996.27	2127.0	997.67	993.02	16 min 9 sec
SAT mut 0.04 cross 1 point 0.7	996.74	1273.0	997.67	995.34	14 min 46 sec
SAT mut 0.04 cross 1 point 0.8	996.74	859.0	997.67	993.02	15 min 5 sec
SAT mut 0.04 cross 1 point 0.9	997.67	1621.0	997.67	997.67	15 min 23 sec
SAT mut 0.04 cross 1 point 1	997.2	953.0	997.67	995.34	15 min 43 sec

Tabulka 50. Test 5

Název souboru:	sat_cross_mut_tour7_2.test uf100-814				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 cross 2 point 0	995.73	1972.0	997.67	990.69	15 min 39 sec
SAT mut 0.04 cross 2 point 0.1	995.34	1533.0	997.67	993.02	13 min 41 sec
SAT mut 0.04 cross 2 point 0.2	996.27	1441.0	997.67	993.02	12 min 24 sec
SAT mut 0.04 cross 2 point 0.3	995.81	1715.0	997.67	993.02	12 min 43 sec
SAT mut 0.04 cross 2 point 0.4	995.81	641.0	997.67	995.34	12 min 55 sec
SAT mut 0.04 cross 2 point 0.5	997.2	1297.0	997.67	995.34	13 min 12 sec
SAT mut 0.04 cross 2 point 0.6	997.67	1211.0	997.67	997.67	13 min 32 sec
SAT mut 0.04 cross 2 point 0.7	996.74	1985.0	997.67	993.02	13 min 38 sec
SAT mut 0.04 cross 2 point 0.8	995.81	2339.0	997.67	990.69	14 min 27 sec
SAT mut 0.04 cross 2 point 0.9	997.2	1839.0	997.67	995.34	16 min 3 sec
SAT mut 0.04 cross 2 point 1	995.81	461.0	997.67	993.02	15 min 14 sec

Tabulka 51. Test 5, pokračování

Název souboru:	sat_cross_mut_tour7_2.test uf100-814				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.04 Uniform cross 0	996.51	1529.0	997.67	995.34	14 min 41 sec
SAT mut 0.04 Uniform cross 0.1	996.27	1493.0	997.67	993.02	12 min 31 sec
SAT mut 0.04 Uniform cross 0.2	995.81	903.0	997.67	993.02	12 min 51 sec
SAT mut 0.04 Uniform cross 0.3	994.88	1893.0	997.67	990.69	13 min 9 sec
SAT mut 0.04 Uniform cross 0.4	996.74	1557.0	997.67	995.34	13 min 23 sec
SAT mut 0.04 Uniform cross 0.5	994.88	1119.0	997.67	993.02	13 min 46 sec
SAT mut 0.04 Uniform cross 0.6	996.27	827.0	997.67	993.02	14 min 14 sec
SAT mut 0.04 Uniform cross 0.7	995.34	1893.0	997.67	993.02	14 min 32 sec
SAT mut 0.04 Uniform cross 0.8	995.81	1437.0	997.67	993.02	15 min 24 sec
SAT mut 0.04 Uniform crosso 0.9	996.27	1141.0	1000.0	993.02	13 min 14 sec
SAT mut 0.04 Uniform cross 1	996.74	1517.0	997.67	995.3	15 min 11 sec

Tabulka 52. Test 5, pokračování

Název souboru:	sat_cross_mut_tour7_2.test				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 cross 1 point 0	996.51	872.0	997.67	990.69	14 min 30 sec
SAT mut 0.02 cross 1 point 0.1	996.74	349.0	997.67	993.02	13 min 57 sec
SAT mut 0.02 cross 1 point 0.2	994.88	165.0	997.67	990.69	12 min 21 sec
SAT mut 0.02 cross 1 point 0.3	995.81	201.0	997.67	993.02	12 min 57 sec
SAT mut 0.02 cross 1 point 0.4	995.34	1141.0	997.67	993.02	15 min 17 sec
SAT mut 0.02 cross 1 point 0.5	995.34	637.0	997.67	993.02	14 min 43 sec
SAT mut 0.02 cross 1 point 0.6	996.74	553.0	997.67	995.34	16 min 59 sec
SAT mut 0.02 cross 1 point 0.7	996.27	581.0	997.67	993.02	16 min 22 sec
SAT mut 0.02 cross 1 point 0.8	994.88	197.0	997.67	993.02	13 min 53 sec
SAT mut 0.02 cross 1 point 0.9	995.81	183.0	997.67	995.34	16 min 9 sec
SAT mut 0.02 cross 1 point 1	995.81	1083.0	997.67	995.34	15 min 58 sec

Tabulka 53. Test 5, pokračování

Název souboru:	sat_cross_mut_tour7_2.test uf100-814				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 cross 2 point 0	996.89	391.0	997.67	995.34	15 min 4 sec
SAT mut 0.02 cross 2 point 0.1	996.27	755.0	997.67	995.34	14 min 3 sec
SAT mut 0.02 cross 2 point 0.2	995.34	161.0	997.67	993.02	15 min 58 sec
SAT mut 0.02 cross 2 point 0.3	996.27	373.0	997.67	993.02	14 min 16 sec
SAT mut 0.02 cross 2 point 0.4	996.74	1095.0	997.67	995.34	13 min 31 sec
SAT mut 0.02 cross 2 point 0.5	995.34	221.0	997.67	993.02	14 min 39 sec
SAT mut 0.02 cross 2 point 0.6	995.34	681.0	997.67	990.69	13 min 5 sec
SAT mut 0.02 cross 2 point 0.7	997.2	611.0	997.67	995.34	14 min 48 sec
SAT mut 0.02 cross 2 point 0.8	995.34	105.0	997.67	990.69	17 min 20 sec
SAT mut 0.02 cross 2 point 0.9	996.74	1009.0	997.67	995.34	17 min 39 sec
SAT mut 0.02 cross 2 point 1	997.2	299.0	997.67	995.34	17 min 29 sec

Tabulka 54. Test 5, pokračování

Název souboru:	sat_cross_mut_tour7_2.testuf100-814				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	nejhorší	
SAT mut 0.02 Uniform cross 0	996.12	776.0	997.67	993.02	17 min 0 sec
SAT mut 0.02 Uniform cross 0.1	997.2	569.0	997.67	995.34	14 min 19 sec
SAT mut 0.02 Uniform cross 0.2	996.27	199.0	997.67	993.02	14 min 34 sec
SAT mut 0.02 Uniform cross 0.3	997.67	363.0	997.67	997.67	17 min 22 sec
SAT mut 0.02 Uniform cross 0.4	997.2	1741.0	1000.0	995.34	13 min 5 sec
SAT mut 0.02 Uniform cross 0.5	996.74	773.0	997.67	995.34	14 min 45 sec
SAT mut 0.02 Uniform cross 0.6	996.27	119.0	997.67	993.02	13 min 43 sec
SAT mut 0.02 Uniform cross 0.7	997.2	987.0	997.67	995.34	13 min 56 sec
SAT mut 0.02 Uniform cross 0.8	995.81	1745.0	997.67	995.34	14 min 55 sec
SAT mut 0.02 Uniform cross 0.9	996.74	311.0	997.67	995.34	18 min 42 sec
SAT mut 0.02 Uniform cross 1	996.74	369.0	997.67	995.34	19 min 52 sec

Tabulka 55. Test 5, pokračování

Instance řešená v tomto testu je opět složitější. Řešení bylo nalezeno pouze použitím uniformního křížení. Vhonější je opět křížení s pravděpodobností 0.02.

Test 6
Tento test je zaměřen na kombinaci křížení a selekce.

Název souboru:	sat_cross_sel_data4.test uf100-0149				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	Nejhorší	
sat simple 0.7 roulette	997.67	2799.0	1000.0	993.02	11 min 31 sec
sat simple 0.7 tour2	999.53	997.0	1000.0	997.67	4 min 34 sec
sat simple 0.7 tour3	999.53	389.0	1000.0	997.67	3 min 12 sec
sat simple 0.7 tour4	998.6	649.0	1000.0	993.02	3 min 26 sec
sat simple 0.7 tour5	998.6	265.0	1000.0	995.34	5 min 17 sec
sat simple 0.7 tour6	996.74	69.0	1000.0	990.69	9 min 33 sec
sat simple 0.7 tour7	998.6	557.0	1000.0	995.34	5 min 37 sec
sat simple 0.7 tour8	998.6	317.0	1000.0	993.02	3 min 2 sec
sat simple 0.7 tour9	999.06	597.0	1000.0	995.34	2 min 42 sec
sat simple 0.7 tour10	999.53	105.0	1000.0	997.67	2 min 28 sec
sat simple 0.7 tour20	994.88	57.0	997.67	993.02	12 min 14 sec
sat simple 0.7 roulette rank	999.53	1319.0	1000.0	997.67	6 min 30 sec
sat simple 0.7 roulette random	990.69	3747.0	993.02	986.04	12 min 37 sec

Tabulka 56. Test 6

Název souboru:	sat_cross_sel_data4.test uf100-0149				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	Nejhorší	
sat 2point 0.7 roulette	996.74	1465.0	997.67	995.34	13 min 5 sec
sat 2point 0.7 tour2	998.13	587.0	1000.0	993.02	5 min 35 sec
sat 2point 0.7 tour3	1000.0	1943.0	1000.0	1000.0	4 min 46 sec
sat 2point 0.7 tour4	999.53	545.0	1000.0	997.67	4 min 40 sec
sat 2point 0.7 tour5	998.6	411.0	1000.0	997.67	7 min 55 sec
sat 2point 0.7 tour6	997.67	459.0	1000.0	995.34	7 min 25 sec
sat 2point 0.7 tour7	997.2	83.0	1000.0	993.02	7 min 8 sec
sat 2point 0.7 tour8	997.67	413.0	1000.0	993.02	7 min 23 sec
sat 2point 0.7 tour9	996.74	989.0	1000.0	993.02	7 min 19 sec
sat 2point 0.7 tour10	999.06	351.0	1000.0	995.34	2 min 55 sec
sat 2point 0.7 tour20	1000.0	445.0	1000.0	1000.0	1 min 7 sec
sat 2point 0.7 roulette rank	998.6	2313.0	1000.0	997.67	11 min 20 sec
sat 2point 0.7 roulette random	991.16	2997.0	995.34	988.37	12 min 40 sec

Tabulka 57. Test 6, pokračování

Název souboru:	sat_cross_sel_data4.test uf100-0149				
Klauzulí	430		Proměnných	100	
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		
	Nejlepších	kroků	nejlepší	Nejhorší	
sat uniform 0.7 roulette	997.67	1681.0	1000.0	995.34	9 min 50 sec
sat uniform 0.7 tour2	998.13	1495.0	1000.0	993.02	6 min 4 sec
sat uniform 0.7 tour3	999.06	929.0	1000.0	995.34	4 min 29 sec
sat uniform 0.7 tour4	1000.0	509.0	1000.0	1000.0	1 min 12 sec
sat uniform 0.7 tour5	999.53	387.0	1000.0	997.67	3 min 19 sec
sat uniform 0.7 tour6	1000.0	819.0	1000.0	1000.0	1 min 57 sec
sat uniform 0.7 tour7	999.06	243.0	1000.0	995.34	2 min 44 sec
sat uniform 0.7 tour8	998.6	351.0	1000.0	995.34	5 min 9 sec
sat uniform 0.7 tour9	999.06	457.0	1000.0	997.67	6 min 11 sec
sat uniform 0.7 tour10	999.53	791.0	1000.0	997.67	4 min 2 sec
sat uniform 0.7 tour20	997.2	93.0	1000.0	993.02	7 min 1 sec
sat uniform 0.7 roulette rank	997.67	1499.0	1000.0	993.02	8 min 19 sec
sat uniform 0.7 roulette random	994.88	2875.0	1000.0	990.69	10 min 46 sec

Tabulka 58. Test 6, pokračování

Opět vychází velmi dobře uniformní křížení a selekce turnajem.

A.4 TSP testy

Pokud není řečeno jinak, je použita mutace s pravděpodobností 0.04.

Test 1
Nyní otestujeme způsoby selekce.

Název souboru:	tsp_sel_methods_data1.test				
Měst	150				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	nejhorší	Čas
tsp sel Roulette	12548.34	X	11496.95	13466.06	34 min 29 sec
tsp sel tour 2	11774.3	X	11351.42	12147.04	33 min 25 sec
tsp sel tour 3	5235.97	X	5147.55	5411.26	33 min 40 sec
tsp sel tour 4	5245.59	X	5094.52	5381.97	32 min 49 sec
tsp sel tour 5	5197.63	X	5096.62	5280.35	37 min 31 sec
tsp sel tour 6	5188.56	X	5070.85	5291.08	38 min 49 sec
tsp sel tour 7	5269.73	X	5165.33	5433.57	37 min 29 sec
tsp sel tour 8	5115.33	X	5009.04	5251.8	35 min 57 sec
tsp sel tour 9	5276.87	X	5074.31	5418.1	38 min 17 sec
tsp sel tour 10	5230.2	X	5165.21	5319.54	36 min 35 sec
tsp sel tour 20	5256.87	X	5136.87	5422.13	36 min 55 sec
tsp sel rank	11493.98	X	10870.68	12175.42	39 min 6 sec
tsp sel random	24966.97	X	24062.64	25829.12	36 min 58 sec

Tabulka 59. Test 1

Test 2
Opakování předchozího testu pro jiná data.

Název souboru:	tsp_sel_methods_data2.test				
Měst	200				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	nejhorší	Čas
tsp sel Roulette	18801.59	X	18533.38	19420.8	54 min 47 sec
tsp sel tour 2	17189.78	X	16492.56	18187.11	54 min 38 sec
tsp sel tour 3	5803.74	X	5684.93	5913.05	51 min 59 sec
tsp sel tour 4	5839.44	X	5773.62	5936.6	53 min 9 sec
tsp sel tour 5	5763.94	X	5531.05	6019.15	51 min 51 sec
tsp sel tour 6	5833.47	X	5768.09	5961.72	52 min 17 sec
tsp sel tour 7	5818.54	X	5671.38	5954.68	48 min 8 sec
tsp sel tour 8	5884.08	X	5754.15	6074.65	50 min 25 sec
tsp sel tour 9	5985.16	X	5863.67	6171.63	57 min 38 sec
tsp sel tour 10	5872.05	X	5816.89	5945.93	60 min 20 sec
tsp sel tour 20	5772.21	X	5695.94	5861.26	48 min 49 sec
tsp sel rank	17237.33	X	16753.27	17787.66	50 min 54 sec
tsp sel random	34210.77	X	32865.28	36368.7	49 min 50 sec

Tabulka 60. Test 2

Test 3
Opakování předchozího testu pro jiná data.

Název souboru:	tsp_sel_methods_data3.test				
Měst	300				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp sel Roulette	33181.88	X	32460.11	33916.99	98 min 28 sec
tsp sel tour 2	27917.47	X	27568.6	28775.14	82 min 12 sec
tsp sel tour 3	7715.78	X	7625.53	7830.6	80 min 37 sec
tsp sel tour 4	7577.2	X	7284.26	7878.4	74 min 34 sec
tsp sel tour 5	7505.78	X	7288.14	7731.74	73 min 6 sec
tsp sel tour 6	7484.17	X	7318.96	7683.22	73 min 9 sec
tsp sel tour 7	7483.26	X	7348.82	7555.14	73 min 10 sec
tsp sel tour 8	7476.71	X	7203.81	7835.56	73 min 13 sec
tsp sel tour 9	7520.6	X	7393.18	7595.48	74 min 39 sec
tsp sel tour 10	7582.3	X	7210.93	7778.7	79 min 48 sec
tsp sel tour 20	7625.68	X	7461.65	7864.1	85 min 26 sec
tsp sel rank	27852.76	X	27453.41	28427.59	83 min 35 sec
tsp sel random	55231.44	X	54177.13	56239.05	84 min 51 sec

Tabulka 61. Test 3

Test 4

Tento test si klade za cíl otestovat různé kombinace křížení a mutace.

Název souboru:	tsp_cross_mut_pack_data1.test				
Měst	150				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross 1 point 0	5788.46	X	5632.67	5929.82	4 min 57 sec
tsp cross 1 point 0.1	5767.29	X	5691.53	5829.69	3 min 45 sec
tsp cross 1 point 0.2	5488.9	X	5332.33	5709.31	11 min 46 sec
tsp cross 1 point 0.3	5525.86	X	5316.22	5661.05	15 min 56 sec
tsp cross 1 point 0.4	5398.01	X	5329.16	5490.28	19 min 55 sec
tsp cross 1 point 0.5	5347.16	X	5252.96	5475.92	23 min 45 sec
tsp cross 1 point 0.6	5464.15	X	5201.59	5626.71	31 min 12 sec
tsp cross 1 point 0.7	5377.43	X	4987.08	5584.98	34 min 16 sec
tsp cross 1 point 0.8	5378.43	X	5191.92	5538.89	38 min 56 sec
tsp cross 1 point 0.9	5319.33	X	5208.72	5567.87	41 min 40 sec
tsp cross 1 point 1	5303.18	X	5144.36	5449.37	46 min 8 sec

Tabulka 62. Test 4

Název souboru:	tsp_cross_mut_pack_data1.test				
Měst	150				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross order1 0	5731.12	X	5526.74	6133.42	4 min 3 sec
tsp cross order1 0.1	5538.71	X	5299.71	5780.23	7 min 32 sec
tsp cross order1 0.2	5378.8	X	5176.52	5524.51	11 min 21 sec
tsp cross order1 0.3	5378.56	X	5259.69	5569.84	15 min 26 sec
tsp cross order1 0.4	5341.69	X	5211.77	5461.12	19 min 16 sec
tsp cross order1 0.5	5215.92	X	5101.29	5272.38	23 min 6 sec
tsp cross order1 0.6	5352.85	X	5242.33	5535.06	29 min 42 sec
tsp cross order1 0.7	5255.19	X	5160.55	5378.68	32 min 38 sec
tsp cross order1 0.8	5166.39	X	5009.3	5270.4	37 min 27 sec
tsp cross order1 0.9	5336.69	X	5127.72	5627.03	39 min 27 sec
tsp cross order1 1	5225.63	X	4997.49	5402.84	45 min 7 sec

Tabulka 63. Test 4, pokračování

Název souboru:	tsp_cross_mut_pack_data1.test				
Měst	150				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp Partially mapped 0	5706.66	X	5559.53	5830.14	4 min 28 sec
tsp Partially mapped 0.1	5579.23	X	5506.81	5694.89	7 min 0 sec
tsp Partially mapped 0.2	5536.65	X	5499.06	5570.09	10 min 39 sec
tsp Partially mapped 0.3	5458.87	X	5337.49	5550.6	14 min 37 sec
tsp Partially mapped 0.4	5507.14	X	5392.16	5602.85	18 min 10 sec
tsp Partially mapped 0.5	5590.99	X	5461.42	5748.72	22 min 1 sec
tsp Partially mapped 0.6	5484.69	X	5401.06	5589.68	25 min 34 sec
tsp Partially mapped 0.7	5483.18	X	5349.34	5639.28	29 min 19 sec
tsp Partially mapped 0.8	5517.0	X	5457.03	5669.46	34 min 16 sec
tsp Partially mapped 0.9	5524.23	X	5370.74	5631.4	37 min 0 sec
tsp Partially mapped 1	5416.68	X	5231.68	5670.55	44 min 17 sec

Tabulka 64. Test 4, pokračování

Název souboru:	tsp_cross_mut_pack_data1.test				
Měst	150				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp mut PMX 0.0	23960.04	X	22067.09	25502.93	37 min 42 sec
tsp mut PMX 0.01	5488.76	X	5384.05	5606.48	36 min 3 sec
tsp mut PMX 0.02	5138.18	X	4954.31	5314.3	35 min 38 sec
tsp mut PMX 0.03	5100.9	X	5045.39	5180.13	36 min 20 sec
tsp mut PMX 0.04	5171.4	X	5051.51	5441.68	33 min 58 sec
tsp mut PMX 0.05	5130.83	X	4965.19	5256.24	35 min 42 sec
tsp mut PMX 0.06	5116.4	X	4967.57	5222.33	35 min 11 sec
tsp mut PMX 0.07	5111.21	X	5027.54	5285.11	38 min 10 sec
tsp mut PMX 0.08	5056.47	X	4920.88	5223.62	41 min 6 sec
tsp mut PMX 0.09	5049.19	X	4956.84	5190.3	42 min 28 sec
tsp mut PMX 0.1	5070.45	X	4887.31	5238.04	40 min 27 sec
tsp mut PMX 0.2	5097.74	X	4983.89	5248.62	41 min 1 sec
tsp mut PMX 0.4	5168.65	X	4953.49	5340.41	39 min 27 sec
tsp mut PMX 0.8	4981.13	X	4811.65	5108.01	32 min 21 sec
tsp mut PMX 1	5951.85	X	5659.2	6262.94	32 min 24 sec

Tabulka 65. Test 4, pokračování

Nejlépe vychází křížení order 1 s pravděpodobností 0.5 - 1, nepříliš dobré výsledky dává partially mapped křížení a křížení v jednom bodě.

Test 5
Opakování předchozího testu pro jiná data.

Název souboru:	tsp_cross_mut_pack_data2.test				
Měst	200				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross 1 point 0	7657.14	X	7420.33	7778.46	6 min 40 sec
tsp cross 1 point 0.1	7599.09	X	7401.96	7986.7	4 min 39 sec
tsp cross 1 point 0.2	6939.04	X	6745.48	7141.18	16 min 49 sec
tsp cross 1 point 0.3	6855.23	X	6610.4	7084.35	22 min 22 sec
tsp cross 1 point 0.4	6828.32	X	6693.83	7011.23	30 min 19 sec
tsp cross 1 point 0.5	6776.28	X	6581.58	7048.27	34 min 42 sec
tsp cross 1 point 0.6	6640.66	X	6485.48	6767.17	42 min 59 sec
tsp cross 1 point 0.7	6604.01	X	6369.4	6893.61	48 min 15 sec
tsp cross 1 point 0.8	6680.31	X	6394.67	6966.15	53 min 17 sec
tsp cross 1 point 0.9	6453.85	X	6184.02	6694.18	60 min 0 sec
tsp cross 1 point 1	6476.34	X	6284.44	6634.91	65 min 46 sec

Tabulka 66. Test 5

Název souboru:	tsp_cross_mut_pack_data2.test				
Měst	200				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross order1 0	7809.62	X	7634.32	8147.98	5 min 7 sec
tsp cross order1 0.1	7175.21	X	7010.87	7330.41	9 min 33 sec
tsp cross order1 0.2	6930.83	X	6768.43	7139.49	15 min 32 sec
tsp cross order1 0.3	6693.59	X	6454.53	6844.83	21 min 8 sec
tsp cross order1 0.4	6658.61	X	6478.48	6857.05	27 min 29 sec
tsp cross order1 0.5	6754.74	X	6642.75	6869.16	33 min 34 sec
tsp cross order1 0.6	6543.33	X	6274.56	6947.17	41 min 17 sec
tsp cross order1 0.7	6529.87	X	6424.02	6713.33	45 min 53 sec
tsp cross order1 0.8	6420.88	X	6227.59	6726.61	52 min 48 sec
tsp cross order1 0.9	6360.81	X	6181.99	6611.36	58 min 56 sec
tsp cross order1 1	6129.8	X	5980.21	6393.65	73 min 52 sec

Tabulka 67. Test 5, pokračování

Název souboru:	tsp_cross_mut_pack_data2.test				
Měst	200				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp Partially mapped 0	7801.44	X	7404.74	7980.01	5 min 13 sec
tsp Partially mapped 0.1	7241.91	X	6900.19	7652.36	11 min 4 sec
tsp Partially mapped 0.2	7128.02	X	6899.43	7479.09	16 min 55 sec
tsp Partially mapped 0.3	7120.78	X	6701.66	7356.76	21 min 28 sec
tsp Partially mapped 0.4	6871.88	X	6782.37	6932.66	29 min 18 sec
tsp Partially mapped 0.5	7111.63	X	6967.92	7394.85	31 min 50 sec
tsp Partially mapped 0.6	6811.15	X	6546.86	7100.12	37 min 47 sec
tsp Partially mapped 0.7	7094.55	X	6914.95	7350.89	39 min 29 sec
tsp Partially mapped 0.8	6806.55	X	6710.05	6872.97	43 min 52 sec
tsp Partially mapped 0.9	6878.34	X	6776.28	6949.59	49 min 24 sec
tsp Partially mapped 1	6976.2	X	6771.39	7176.7	61 min 23 sec

Tabulka 68. Test 5, pokračování

Název souboru:	tsp_cross_mut_pack_data2.test				
Měst	200				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp mut PMX 0.0	35048.83	X	32806.19	36692.52	47 min 12 sec
tsp mut PMX 0.01	6945.93	X	6735.41	7058.21	46 min 48 sec
tsp mut PMX 0.02	6253.07	X	6185.2	6383.8	47 min 27 sec
tsp mut PMX 0.03	6014.78	X	5849.16	6288.8	47 min 39 sec
tsp mut PMX 0.04	5891.75	X	5738.53	5969.12	47 min 51 sec
tsp mut PMX 0.05	5885.04	X	5768.39	5997.8	47 min 41 sec
tsp mut PMX 0.06	5752.41	X	5603.56	5915.5	47 min 27 sec
tsp mut PMX 0.07	5797.95	X	5671.6	6001.06	46 min 38 sec
tsp mut PMX 0.08	5797.16	X	5738.52	5852.75	46 min 2 sec
tsp mut PMX 0.09	5891.59	X	5750.87	6003.38	45 min 54 sec
tsp mut PMX 0.1	5819.12	X	5773.88	5865.79	45 min 50 sec
tsp mut PMX 0.2	5857.02	X	5722.88	6018.31	45 min 41 sec
tsp mut PMX 0.4	5831.9	X	5692.06	5926.52	51 min 56 sec
tsp mut PMX 0.8	5837.1	X	5544.94	6079.61	48 min 16 sec
tsp mut PMX 1	8121.48	X	7623.32	8360.98	49 min 2 sec

Tabulka 69. Test 5, pokračování

Nejlépe opět vychází křížení order 1 s pravděpodobností 0.9 - 1. Mutace vychází nejlépe s hodnotami pravděpodobností 0.06 - 0.08.

Test 6
Opakování předchozího testu pro jiná data.

Název souboru:	tsp_cross_mut_pack_data3.test				
Měst	300				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross 1 point 0	12946.14	X	12285.87	13466.91	7 min 54 sec
tsp cross 1 point 0.1	12914.67	X	12619.67	13200.26	5 min 33 sec
tsp cross 1 point 0.2	10840.48	X	10636.32	11013.57	24 min 44 sec
tsp cross 1 point 0.3	10626.06	X	10134.47	10870.34	34 min 42 sec
tsp cross 1 point 0.4	10360.82	X	10101.84	10656.32	44 min 48 sec
tsp cross 1 point 0.5	10042.66	X	9748.1	10555.1	55 min 27 sec
tsp cross 1 point 0.6	10066.15	X	9894.14	10258.91	67 min 21 sec
tsp cross 1 point 0.7	9796.22	X	9494.29	9930.06	80 min 58 sec
tsp cross 1 point 0.8	9585.72	X	9016.38	9905.28	90 min 53 sec
tsp cross 1 point 0.9	9661.97	X	9157.81	9947.01	102 min 11 sec
tsp cross 1 point 1	9449.01	X	9217.39	9583.25	113 min 15 sec

Tabulka 70. Test 6

Název souboru:	tsp_cross_mut_pack_data3.test				
Měst	300				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp cross order1 0	12801.94	X	12392.74	13075.26	5 min 45 sec
tsp cross order1 0.1	11453.3	X	10970.78	11840.74	13 min 17 sec
tsp cross order1 0.2	10818.74	X	10381.05	11252.67	22 min 6 sec
tsp cross order1 0.3	10337.99	X	9895.83	10629.51	32 min 7 sec
tsp cross order1 0.4	9893.7	X	9659.47	10171.94	39 min 33 sec
tsp cross order1 0.5	9720.31	X	9561.44	9951.11	49 min 44 sec
tsp cross order1 0.6	9632.09	X	9180.65	10072.25	57 min 41 sec
tsp cross order1 0.7	9321.67	X	9087.11	9474.75	65 min 56 sec
tsp cross order1 0.8	9061.4	X	8683.49	9271.3	74 min 21 sec
tsp cross order1 0.9	9102.96	X	8684.81	9437.17	82 min 49 sec
tsp cross order1 1	8919.73	X	8555.74	9428.59	91 min 34 sec

Tabulka 71. Test 6, pokračování

Název souboru:	tsp_cross_mut_pack_data3.test				
Měst	300				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp Partially mapped 0	12822.67	X	12284.23	13154.08	6 min 3 sec
tsp Partially mapped 0.1	11754.13	X	11491.24	11925.79	13 min 11 sec
tsp Partially mapped 0.2	11257.39	X	10692.78	11872.14	21 min 19 sec
tsp Partially mapped 0.3	10981.46	X	10753.14	11430.97	29 min 13 sec
tsp Partially mapped 0.4	10664.58	X	10057.94	11481.53	37 min 24 sec
tsp Partially mapped 0.5	10400.83	X	10190.99	10848.03	45 min 2 sec
tsp Partially mapped 0.6	10482.88	X	10281.85	10731.27	52 min 35 sec
tsp Partially mapped 0.7	10430.26	X	10014.18	10727.91	60 min 56 sec
tsp Partially mapped 0.8	10323.59	X	10067.54	10507.43	69 min 13 sec
tsp Partially mapped 0.9	10378.17	X	10174.12	10594.37	77 min 23 sec
tsp Partially mapped 1	10101.77	X	9804.36	10412.51	85 min 28 sec

Tabulka 72. Test 6, pokračování

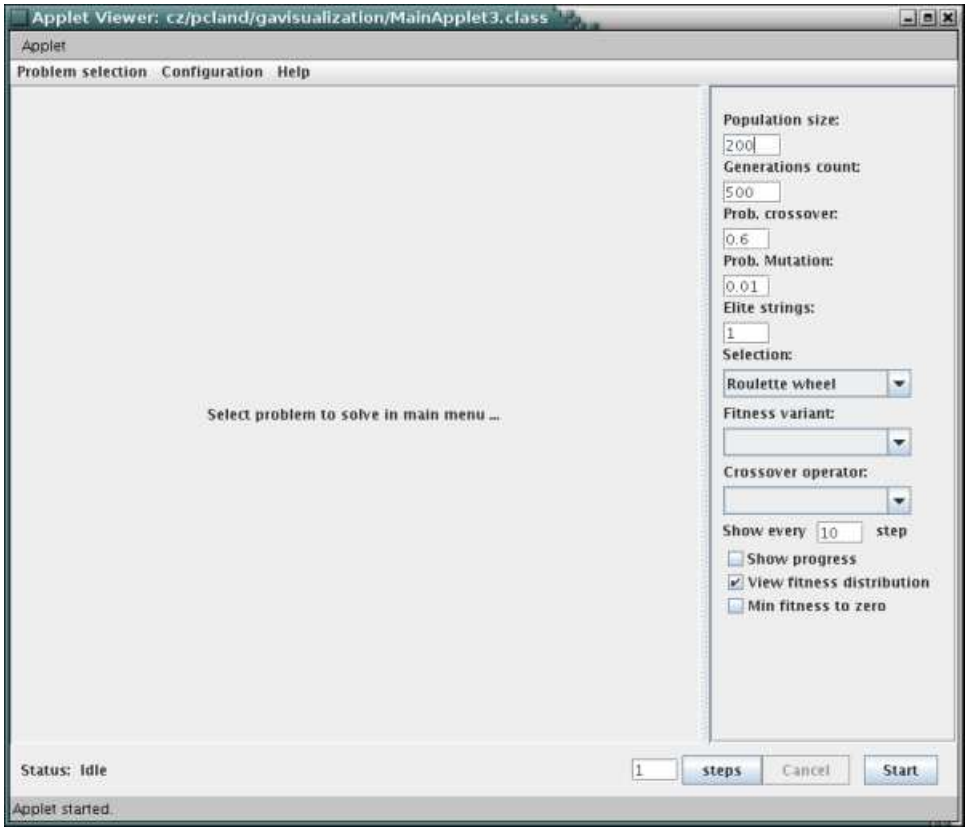
Název souboru:	tsp_cross_mut_pack_data3.test				
Měst	300				
Popis testu	Průměr_z_5_testů		Z_5_testů_řešení		Celkem
	Nejlepších	kroků	Nejlepší	Nejhorší	Čas
tsp mut PMX 0.0	54635.62	X	50249.75	56393.12	69 min 51 sec
tsp mut PMX 0.01	10522.26	X	10250.06	10956.22	70 min 43 sec
tsp mut PMX 0.02	8976.83	X	8665.46	9382.5	71 min 37 sec
tsp mut PMX 0.03	8131.69	X	7961.85	8323.22	71 min 38 sec
tsp mut PMX 0.04	7825.73	X	7648.05	8131.85	71 min 42 sec
tsp mut PMX 0.05	7549.04	X	7407.77	7730.14	72 min 1 sec
tsp mut PMX 0.06	7530.1	X	7410.39	7618.68	71 min 11 sec
tsp mut PMX 0.07	7272.2	X	7198.02	7391.01	72 min 0 sec
tsp mut PMX 0.08	7190.79	X	7108.18	7323.67	71 min 13 sec
tsp mut PMX 0.09	7223.51	X	7162.42	7351.22	69 min 18 sec
tsp mut PMX 0.1	7193.0	X	7083.64	7285.22	69 min 30 sec
tsp mut PMX 0.2	7156.46	X	6987.25	7383.12	69 min 27 sec
tsp mut PMX 0.4	7149.04	X	6915.52	7261.73	69 min 48 sec
tsp mut PMX 0.8	7217.56	X	7067.11	7632.0	73 min 18 sec
tsp mut PMX 1	14527.41	X	14273.34	14932.28	73 min 28 sec

Tabulka 73. Test 6, pokračování

Opět vyšlo z testu nejlépe *"order 1"* křížení a velké hodnoty mutace. Tyto hodnoty se mohou zdát opravdu vysoké, je ale třeba uvědomit si, že tato pravděpodobnost se vztahuje k celému řetězci a né pouze k jednomu genu, jako je tomu při reprezentaci binárními řetězci. Pravděpodobnost 1/n, kde n je délka řetězce, představuje přibližně jednu změnu na celý řetězec (u binárního řetězce). V případě TSP a permutačního kódování odpovídá přibližně jedné změně na n řetězců.

Dodatek B Uživatelská příručka

Na obrázku se nachází úvodní okno aplikace. Horní část aplikace obsahuje hlavní menu.



Obrázek 8. GUI po spuštění apletu

Jednotlivé položky menu jsou:

- Problem selection - slouží k výběru problému, který bude řešen genetickým algoritmem. Volby jsou následující:
  - → Knap Sack (problém batohu)
  - → SAT (problém splnitelnosti booleovské formule)
  - → Function maximum (hledání maxima funkce)
  - → Travelling salesman (problém obchodního cestujícího)
- Configuration - slouží k nastavení programu a spoučtění skriptů. Možné volby jsou:
  - → Visualize fitness - na vizualizačním panelu zobrazí v grafu hodnoty fitness.
  - → Visualize solution evaluation - na vizualizačním panelu zobrazí v grafu ohodnocení hodnotící funkce. To se může od fitness výrazně lišit. Například hodnotící funkce u TSP předá jako ohodnocení délku cesty obchodního cestujícího mezi městy, ta ale nemůže reprezentovat šance k reprodukci, naopak, čím delší cesta je, tím jsou šance menší.

→ Hide configuration - schová nebo zobrazí panel s nastavením genetického algoritmu.

→ Run script - spustí uživatelský skript.

→ Script output - v novém okně zobrazí výstup ze skriptu.

- Help

  → About ... - zobrazí dialog "O programu ...".

V pravé části aplikace se nachází nastavení genetického algoritmu, které je společné pro všechny řešené problémy.

Jednotlivé položky nastavení:

- Population size (velikost populace) - slouží k nastavení velikosti populace genetického algoritmu.
- Generation count (počet generací) - určuje počet generací běhu genetického algoritmu.
- Prob. crossover (pravděpodobnost křížení) - nastavení pravděpodobnosti, se kterou dojde ke křížení náhodné dvojice po provedení selekce.
- Prob. mutation (pravděpodobnost mutace) - nastavení pravděpodobnosti, se kterou bude docházet k mutaci.
- Elite strings (elitních řetězců) - počet elitních řetězců. Elitní řetězce jsou kopírovány beze změny do příští generace.
- Selection (selekce) - volba metody výběru jedinců. Volba je nezávislá na zvoleném problému.
- Fitness variant (varianta hodnotící funkce) - volba z možných hodnotících funkcí pro zvolený problém.
- Crossover operator (operátor křížení) - volba operátoru křížení pro daný problém.
- Show every (ukázat každý) - touto volbou uživatel upřesní četnost informací zaznamenaných v průběhu algoritmu. Důležité ukazatele běhu algoritmu budou zaznamenávány a zobrazeny vždy po uběhnutí zvoleného počtu kroků (generací).
- View fitness distribution (zobrazit rozložení fitness) - pokud je tato volba aktivní, bude ve výsledném grafu graficky naznačeno také rozložení jedinců pro různé hodnoty fitness. Čím více jedinců je obsaženo v rozmezí hodnot fitness v dané generaci, tím tmavší barva je použita pro vykreslení podkladu.
- Min. fitness to zero (minimální fitness nastav na nulu) - provede odečtení nejmenší

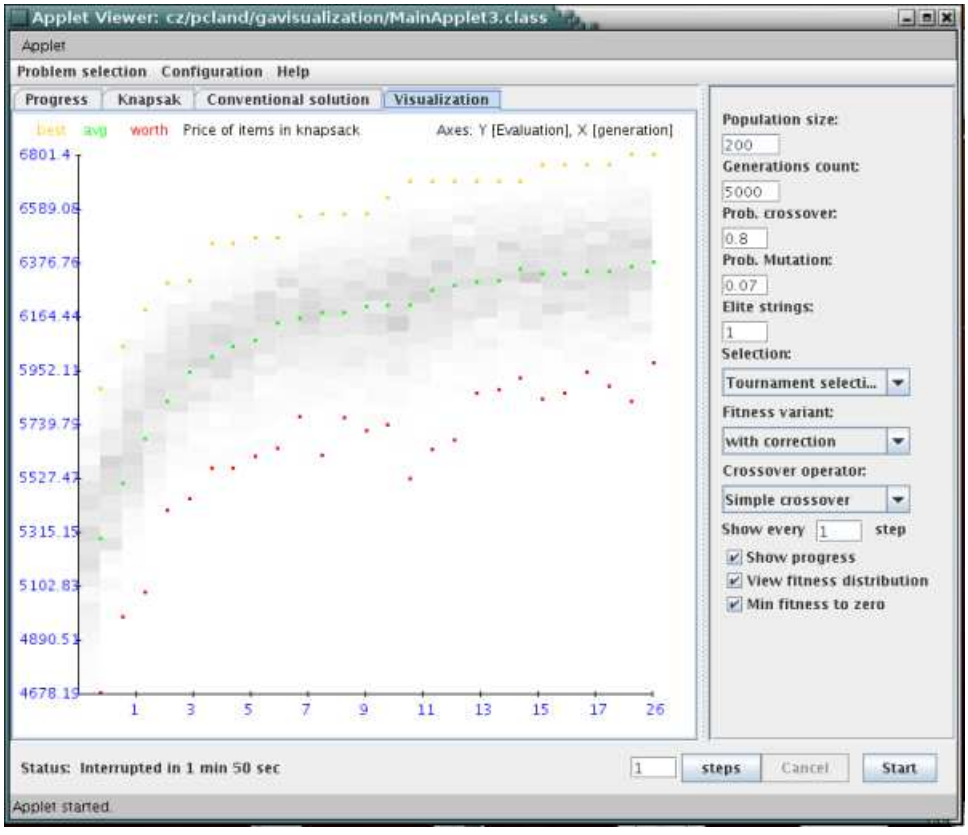
hodnoty fitness, která se v populaci vyskytuje, od každého jedince. Výsledkem je, že jedinec s původně nejnižší hodnotou fitness, bude mít novou hodnotu fitness nulovou.

- Show progress - při této volbě budou aktualizovány vizualizační komponenty v průběhu algoritmu. Pokud je volba neaktivní, budou komponenty aktualizovány až po průběhu všech generací.

V dolní části aplikace se nachází dvě tlačítka. Tlačítko s nápisem start spouští běh algoritmu. Tlačítko steps spustí běh algoritmu pro zvolený počet kroků (generací). Pokud je možné pokračovat v ukončeném běhu, algoritmus bude pokračovat v řešení předchozího zadání.

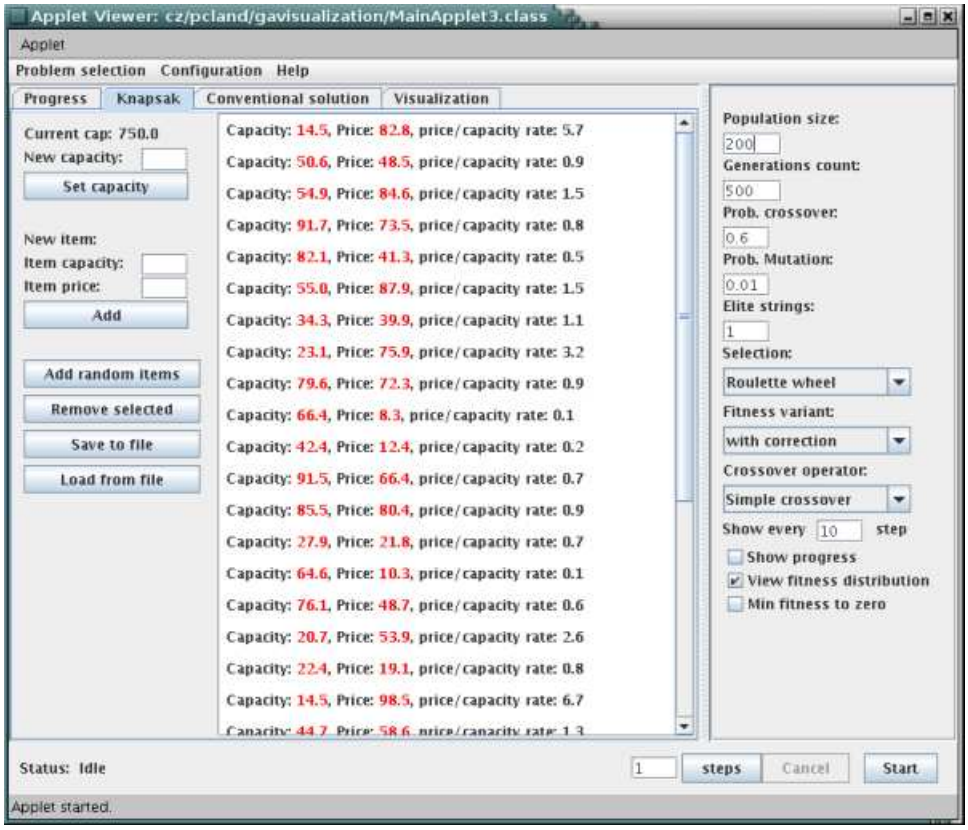
Pro řešení problému genetickým algoritmem musí uživatel nejprve zvolit řešený problém v hlavním menu kliknutím na příslušný problém v položce menu Problem selection. Po načtení problému se zobrazí panel, který obsahuje další nastavení pro řešení problému a vizualizaci.

Pro všechny problémy je možné zobrazit průběh genetického algoritmu na vizualizačním panelu.



Obrázek 9. Průběh vizualizace

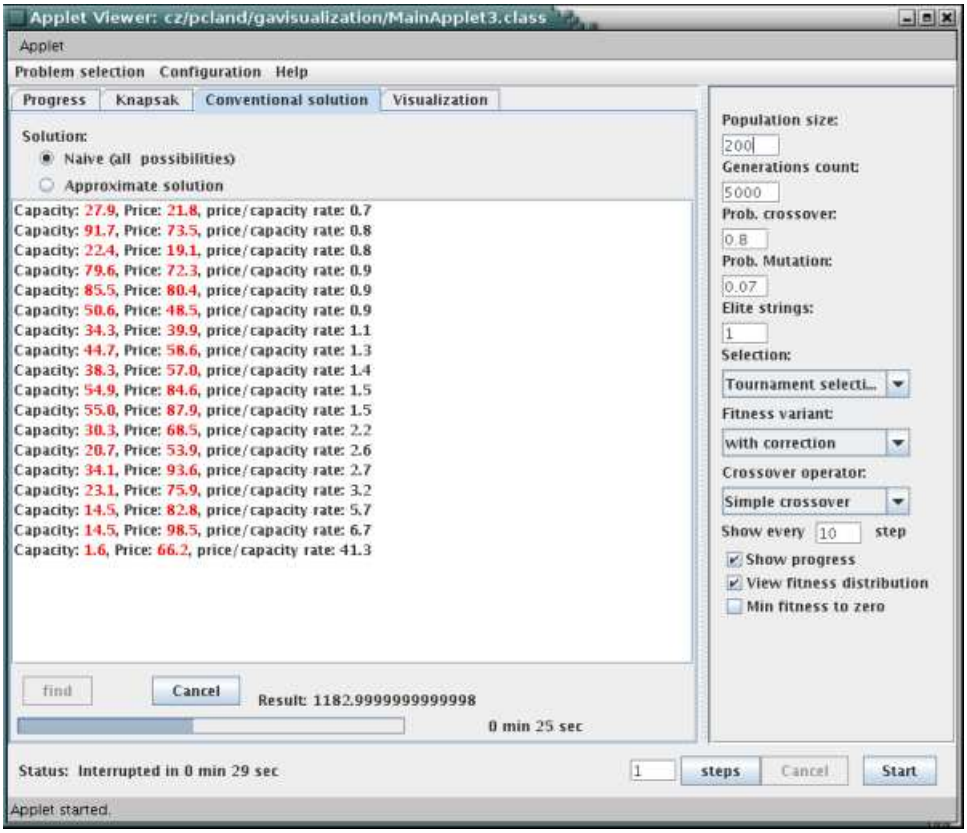
Nastavení parametrů pro problém batohu



Obrázek 10. Nastavení parametrů problému batohu

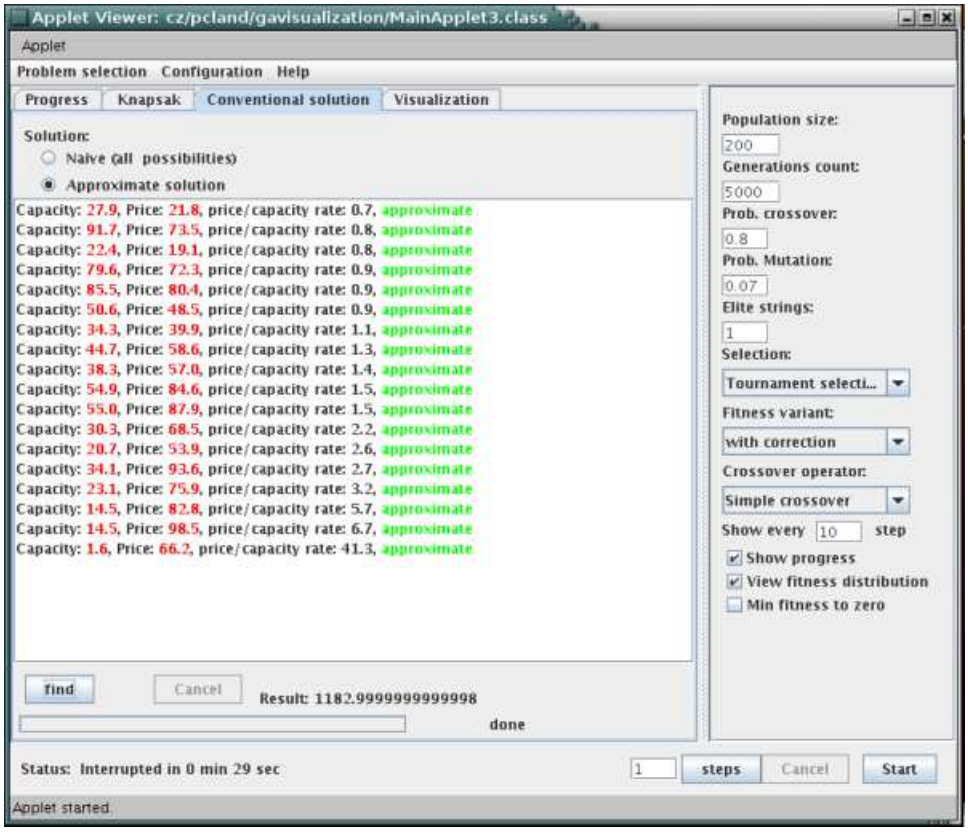
Aplikace nabízí nastavení předmětů, mezi kterými bude algoritmus hledat nejlepší kombinaci. Jednotlivé volby jsou:

- Set capacity - nastaví objem batohu.
- Add - přidá předmět s udanými parametry.
- Add random items - přidá zvolený počet náhodných předmětů.
- Remove selected - odebere vybrané předměty. Pro výběr předmětu je nutné označit zvolené předměty v seznamu všech předmětů.
- Load from file - načte předměty a kapacitu batohu ze souboru.
- Save to file - uloží předměty a kapacitu batohu do souboru.



Obrázek 11. Konvenční řešení problému batohu

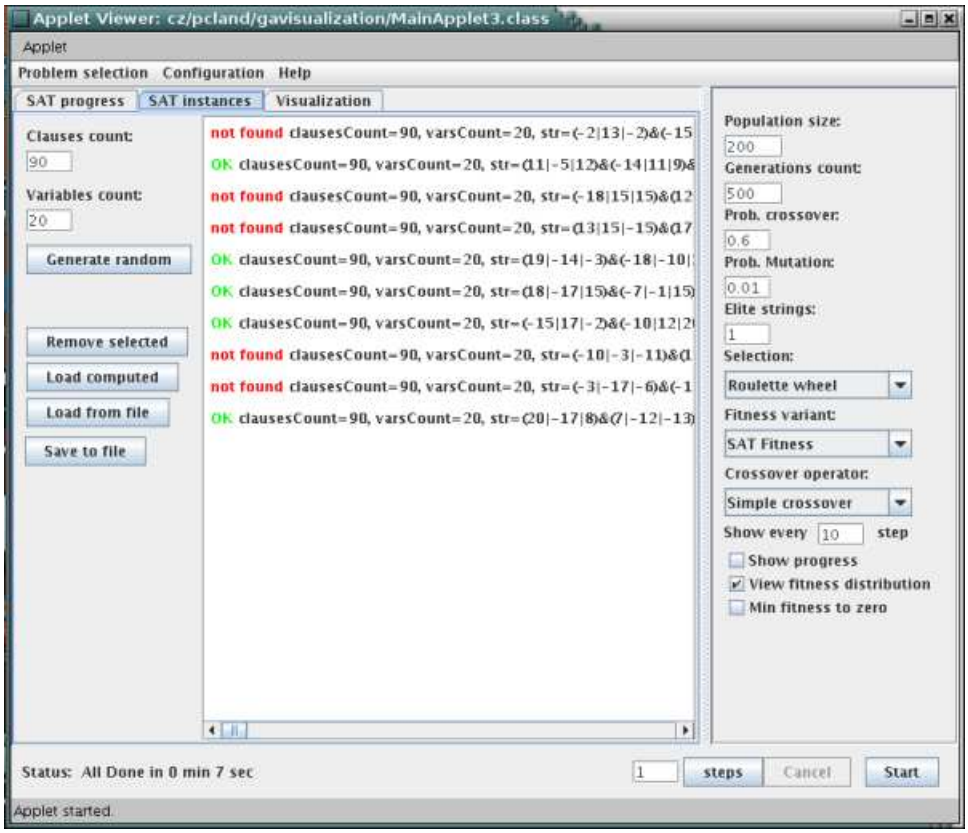
Aplikace umožňuje řešení problému batohu konvenčním způsobem, přepnutím záložky Conventional solution. Zde je třeba vybrat volbu Naive, poté budou testovány všechny kombinace předmětů. V dolní části je zobrazen dosavadní pokrok a čas, který je odhadován do dokončení výpočtu. Volba approximate solution omezí výpočet pouze na jednoduchou heuristiku. Provede seřazení předmětů podle poměru cena/objem a postupně přidává předměty do batohu. Začíná s předmětem, který má nejvyšší poměr a pokračuje, dokud není batoh zaplněn.



Obrázek 12. Přibližné řešení problému batohu

Spuštění genetického algoritmu se provede stiskem tlačítka Start. Vývoj populace je možné sledovat v průběhu algorimu na vizualizačním panelu.

SAT problém



Obrázek 13. Nastavení pro SAT problém

Volby pro SAT instance jsou tyto:

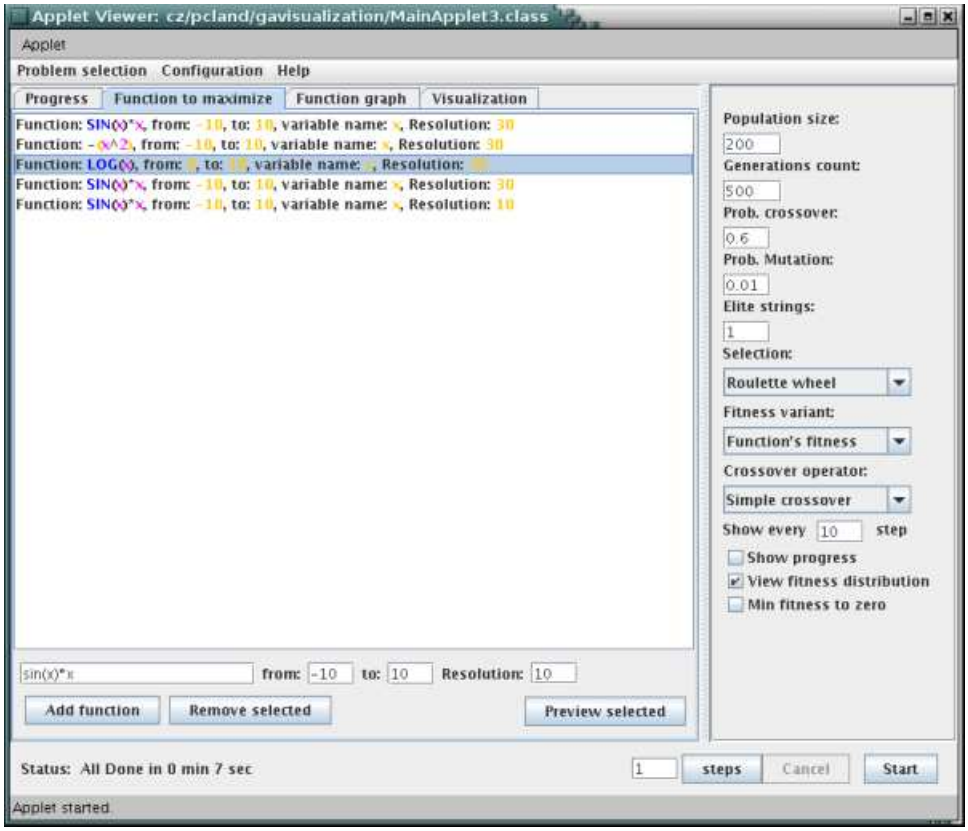
- Generate random - vygeneruje náhodnou instanci se zadanými parametry pomocí

interního generátoru.

- Remove selected - Odebere SAT instance, které jsou označeny v seznamu.
- Load computed - Na vizualizačním panelu zobrazí průběh výpočtu označené instance.
- Load from file - Načte instanci ze souboru.
- Save to file - Uloží instanci do souboru.

Při stisku tlačítka Start se spustí genetický algoritmus pro všechny instance v seznamu (pokud není žádná vybrána). Pokud jsou označeny konkrétní instance, budou řešeny pouze tyto.

Maximalizace funkce



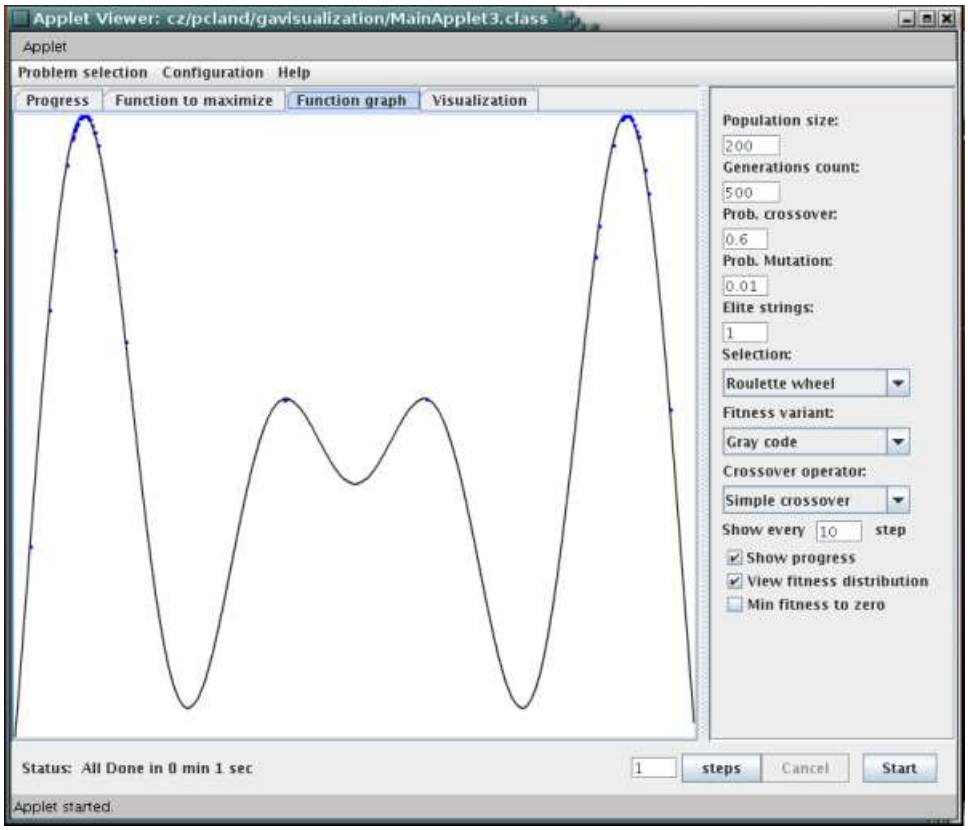
Obrázek 14. Nastavení pro maximalizaci funkce

Aplikace umožňuje uživateli přidat vlastní funkci. Funkci je nutné zapsat do pole, které je k tomu určeno (viz. obrázek, proměnná musí být x), dále je třeba zapsat interval (from, to) na kterém bude maximum hledáno a rozlišení, které je použito pro diskretizaci intervalu prohledávání. Maximální hodnota rozlišení je 30 (bitů).

Jednotlivé volby jsou:

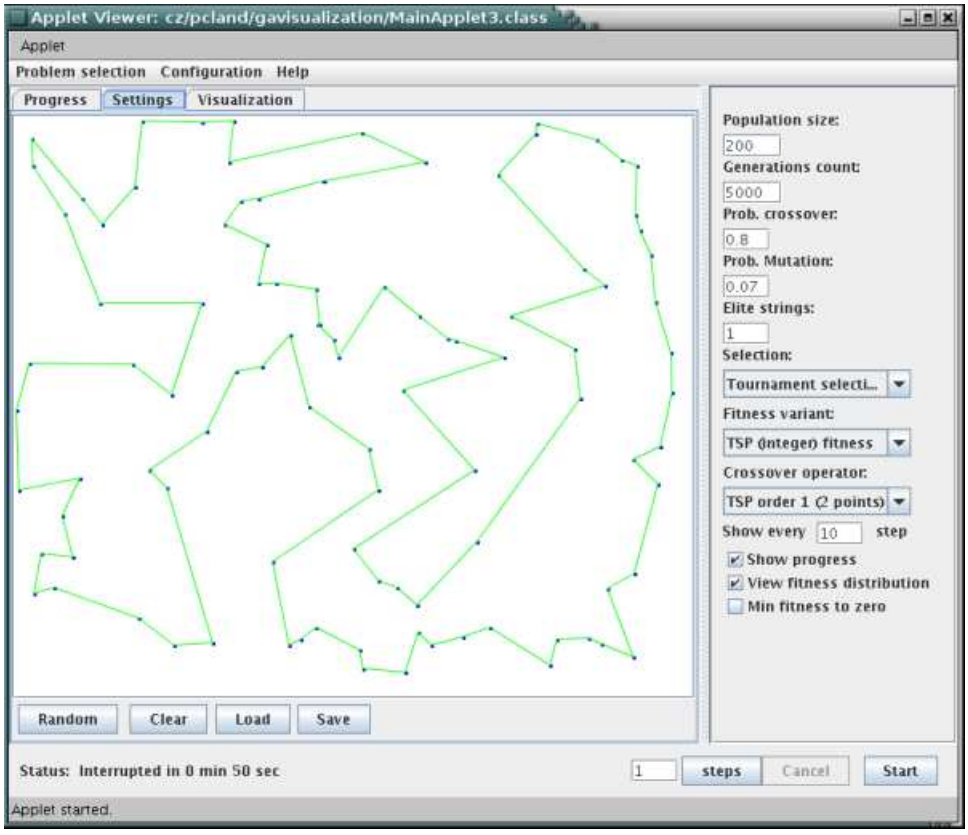
- Add function - přidá do seznamu zadanou funkci.
- Remove selected - odstraní ze seznamu zvolené funkce.
- Preview selected - vykreslí zvolenou funkci.

Po stitsku tlačítka Start se spustí genetický algoritmus pro zvolenou funkci. Průběh je možné sledovat po aktivaci záložky Function graph.



Obrázek 15. Průběh hledání maxima funkce

Problém obchodního cestujícího (TSP)



Obrázek 16. Průběh řešení TSP

Možné jsou tyto volby:

- Random (náhodné zadání) - náhodně rozmístí zvolený počet měst.
- Clear (smazat) - odstraní všechna města.
- Load (načíst) - načte zadání ze souboru.
- Save (uložit) - uloží současné zadání do souboru.

Po stisku tlačítka Start je spuštěn genetický algoritmus. Průběh řešení je možné pozorovat, pokud je volba Show progress aktivní.

Dodatek C Obsah CD

INSTALL	pokyny pro instalaci a spuštění programu
README	obsah cd, další informace
applet.html	HTML soubor obsahující aplet
build	přeložené soubory programu
dist	přeložnený program v jar archívu
html	HTML soubory
doc	dokumentace ve formátu JavaDoc
abstract_cz	annotace česky
abstract_en	anotace anglicky
index.html	úvodní stránka, obsahuje podstatné odkazy
src	zdrojové kódy programu
text	text práce ve formátu pdf
source	text ve formátu $\TeX_{\text{MACS}}$
tests	soubory s testy