Zpráva k semestrální práci z předmětu BI-ZUM

Tomáš Vahalík
vahalto1@fit.cvut.cz
14.5.2018

1. Abstrakt

Zadání bylo pomocí genetického algoritmu nalézt minimální vrcholové pokrytí grafu. Kromě základních operátorů křížení, mutace a selekce jsem přidal i lokální prohledávání, katastrofu, ořezávání listů a operátor opravy, aby algoritmus generoval pouze validní řešení. Z použitých technik měl na výsledek největší vliv asi opravný operátor a předpřipravení problému ořezáváním listů.

Lokální prohledávání probíhá formou simulovaného žíhání. Vždy než je jedinec přidán do nové populace, vygeneruje si několik bodů ze svého okolí (mutací) a když najde lepšího, přidá se tento jedinec do nové populace místo něho. Pokud lepšího jedince nenajde, s určitou pravděpodobností se přidá i jedinec horší. Jaká tato pravděpodobnost bude a kolik bodů z okolí se maximálně generuje, určuje teplota, kterou postupně redukuje hlavní evoluční cyklus.

2. Parametry algoritmu

Jedinec je zakódován jako binární vektor, jehož délka se rovná počtu vrcholů v grafu. 1 je na těch místech, kde je ve výsledku příslušný vrchol označen.

Vzhledem k tomu že algoritmus generuje pouze validní řešení, není potřeba do výpočtu fitness zakomponovat nepokryté hrany. Fitness se tedy rovná počtu vrcholů, které jsou neoznačené.

Pravděpodobnost křížení a mutace jsem nechal tak, jak bylo nastaveno v počáteční šabloně (30% a 2 %). Čím delší evoluce, tím lepší řešení vygeneruje. Nejvíce jsem měl trpělivost na 3000 generací.

3. Operátor mutace

Použil jsem bit-flip mutaci. Vektor jedince se postupně prochází a každý bit se prohodí s pravděpodobností p.

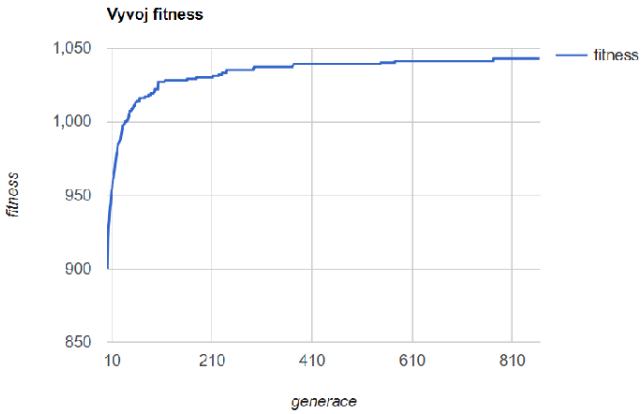
4. Operátor křížení

Křížení je jednobodové, přičemž bod křížení se vybírá náhodně.

5. Selekce

Selekce probíhá turnajovou formou. Funkce dostane argumentem kolik nových jedinců má vybrat. Zvolí tedy 30 náhodných kandidátů a vybere jednoho z nich. Toto se opakuje tak dlouho, dokud není vybráno požadované množství jedinců.

6. Vývoj hodnoty fitness



Obrázek 1: Graf vývoje fitness

Na grafu je vidět, že fitness roste velmi rychle v prvních generacích, poté se ale řešení zlepšuje jen pomalu (asi tak 1 vrchol za 100 generací). Toto značí, že můj algoritmus předčasně konverguje (i při použití katastrofy). Dalo by se tomu předejít použitím některých metod nichingu (např. ostrovního modelu).

7. Shrnutí a výsledky

Podařilo se mi nalézt řešení, kde je 1545 vrcholů, což je od globálního optima sice ještě kus, ale jsem s výsledkem celkem spokojený. Od globálního optima to sice ještě kus je, ale myslím, že výsledek to není špatný.

Jak jsem již psal výše, algoritmus předčasně konverguje a dal by se ještě vylepšit.

Semestrální práce se mi nicméně líbila, šablona v Javě byla dobrá hlavně pro to, že jsem mohl na obrázku sledovat, jak algoritmus pracuje (platí i pro minulé úlohy jako na hledání nejkratších cest. Bez vizualizace by to byla nuda.) Také jsem nemusel řešit spoustu programování okolo (Node, Edge a podpůrné metody na práci s grafem...) ale mohl jsem se soustředit na programování genetického algoritmu.