Zpráva k semestrální práci z předmětu BI-ZUM

Ondřej Kvapil
kvapiond@fit.cvut.cz

16. května 2018

Abstrakt

Implementoval jsem evoluční algoritmus pro aproximaci řešení problému vrcholového pokrytí grafu za pomoci následujících pokročilých evolučních technik: simulované žíhání (přibližná změna průměru fitness ovlivňuje pravděpodobnost mutace), heuristická inicializace (lépe připojené uzly mají vyšší prioritu), vícevláknový běh (křížení a mutace jedinců je distribuovaná do tří vláken) a katastrofa (čistě náhodná, 0.2% šance na náhradu 50% jedinců každou generaci). Fitness se počítá jako záporná hodnota počtu pokrytých uzlů. Výsledek mojí práce se přibližuje globálnímu optimu na menších grafech (3000 uzlů a níž) s přiměřeným počtem generací.

1 Parametry algoritmu

Jedinec je reprezentován binárním vektorem rozhodujícím o zahrnutí příslušného uzlu do řešení. Algoritmus typicky vyžaduje mezi 3500 a 4000 generací než dospěje ke slušnému výsledku. Pravděpodobnost mutace je ze začátku typicky nastavena na 25%, avšak díky implementaci operátoru mutace (viz níže) tato hodnota nehraje velkou roli. Pravděpodobnost křížení je 30%.

2 Operátor mutace

Operátor stochasticky přeskakuje některé indexy v binárním vektoru jedince, jen u těch, kterými opravdu projde, přepne hodnotu s pravděpodobností rovnou zadanému parametru. Tento parametr si ale algoritmus mění sám v průběhu - zvyšuje a snižuje ho v reakci na změny průměrné fitness v populaci. Takto je implementováno simulované žíhání - při malých změnách fitness se pravděpodobnost mutace zvyšuje.

3 Operátor křížení

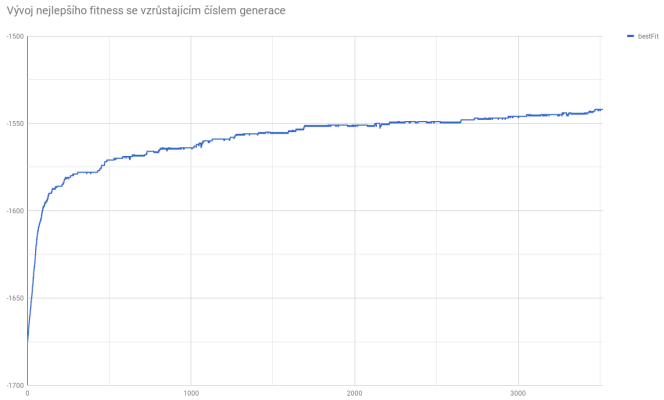
Křížení je n-bodové. Výběr obou rodičů probíhá náhodně. Jeden z nich pochází vždy z nejlepšího (jednoho) procenta populace, druhý se vybere náhodně. Asexuální rozmnožování je povoleno.

4 Selekce

Do další generace postupují všichni jedinci. Tvorba nového jedince je pomalá a zbytečná.

5 Vývoj hodnoty fitness

Následuje ukázka běhu programu na šabloně číslo 2. Algoritmus dosáhl po 130 sekundách a 3518 generacích nejlepší hodnoty fitness -1542, tedy 1542 pokrytých uzlů. Globální optimum pro tento případ je 1502 uzlů. Velikost populace byla 502 jedinců, počáteční pravděpodobnost mutace 20%, křížení 30%. Nejlepší výsledek, kterého se podařilo dosáhnout, bylo 1534 uzlů.



Obrázek 1: Graf vývoje fitness

Jak 1 ukazuje, fitness se vyvíjí téměř logaritmicky. Po cestě jsou patrné body, kde vývoj stagnoval. Z těchto míst ale algoritmus dostane vysoká mutace a/nebo katastrofa. Rozhodně je zde místo pro zlepšení, dobře implementované odřezávání listů by náběh značně zrychlilo.

6 Shrnutí a výsledky

Algoritmus úspěšně aproximuje přijatelné řešení. Implementace zmíněných technik byla zdařilá, některé jiné se ale světla konce semestru nedočkaly. Deterministic crowding navzdory očekáváním nepřineslo užitečné výsledky. Iterativní odřezávání listů bylo implementačně náročnější, než jiné techniky, a bylo tedy během vývoje nahrazeno. Vícevláknový běh nakonec funguje, během práce se s ním ale objevily velké problémy. A konečně na celé práci jsem strávil více času, než by mi bylo milé přiznat, protože mi přibližně osm hodin trvalo přijít na nedostatek v prostém operátoru mutace.

Nejlepším fitness pro šablonu číslo 3 byla hodnota -5987. Doladěním různých parametrů algoritmu by se jistě dala ještě pozvednout.

Co se týče nápadadů a připomínek, mám snad jen jednu. Java šablony ZUM jsou zastaralé, nepraktické a zkopmilované. Chybí k nim dokumentace. Spoléhají navíc na NetBeans, které častokrát nadělaly více škody, než užitku. Připomínám, že všichni studenti FIT mají školní licenci na IntelliJ IDEA, která obsahuje i schopný disassembler. Ten mi během práce nejednou pomohl. Codebase předmětu ZUM je rozhodně třeba osvěžit.