universität
innsbruck

Fundamentals of (Nature-Inspired) Optimization

Project Description

Dr.-Ing. Fedor Smirnov

Outline

- Overview
- Optimization problems
- Google Guice
- Code style
- Report

Overview

Task	:	Implementing 2 optimizers for 2 given problems
Environment	:	Opt4J (Java)
Reference	:	EA-based optimization (see Repository)
Deliverables	:	Project code + report

- The optimizers are chosen during the project registration.
- Of the two optimizers, one is based on swarm intelligence and the other on particle swarm optimization.

Optimization Problems – DTLZ

- Actually a benchmark for multi-objective optimizers [1]
- Continuous optimization problem
- Recommended to be tackled first
- Recommended to be approached using the PSO-based optimizer

Optimization Problems – TSP

- Standard traveling salesman problem
- City map created in a randomized process (seeded random)
- Example of graph- and permutation-based problem
- Recommended to be addressed by the SI approach
- The jung library (`http://jung.sourceforge.net/`) is nice for working with graphs

Google Guice

The software architecture of Opt4J is entirely based on dynamic dependency injection, which is implemented using the **Google Guice framework**.

- Guice is awesome and enables you to significantly improve the way you write code
- To provide an oversimplified summary, Guice enables you to...
  - ...program against interfaces instead of against concrete classes.
  - ...delegate the construction of objects to Guice.
  - ...(dynamically) configure the interface-to-class binding.
- However, working with Guice is a little weird at the start and it takes a while to see and appreciate the possibilities it offers
- My guess is that Guice will be the main difficult during the familiarization with the structure of Opt4J
- A comprehensive explanation of Guice’s functionality can be found in Guice Playlist and Brief Guice Video.

Time Frame

The project is to be completed within 4 weeks. A possible division of time would be:

- **Week 1:** Familiarization with the architecture of Opt4J. Plan for the implementation of the algorithms (how will it work? which components need to be replaced?)
- **Week 2:** Implementing the PSO algorithm for the DTLZ problem.
- **Week 3:** Implementing the SI algorithm for the TSP problem.
- **Week 4:** Incorporating project feedback, writing the report.

Report

- A short summary of the main problems which you encountered and a general outline of the resolution
- Performance comparison to the EA based solution
- No mandatory length. The report should however provide a clear and correct assessment of the performance of the implemented algorithms.

Code Style

Apart from the functionality of the submitted solution and the report, the code style of the submitted code will influence the project grade.

Positive features of code include:

- Comments (class and method heads, parameters, **authorship**)
- Proper architecture (e.g., inheritance instead of copy-pasting code)
- Tests (the code coverage can be measured, e.g., using Jacoco)

Literature References I

Deb, Kalyanmoy et al. (2002). “Scalable multi-objective optimization test problems”. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600)*. Vol. 1. IEEE, pp. 825–830.

Thank you for your attention!

Dr.-Ing. Fedor Smirnov