1 Organisation | Prüfungsleistungen

Organisation

- **MGH-TINF19 | 13.04.2022 09:00 - 12:15 | Online**
  Swarm Intelligence | Foundations + Boids
  Particle Swarm Optimization | Foundations + Implementation
  `https://meet.goto.com/645075949`

  13:15 - 16:30 | Case study + Q&A

- **MGH-TINF19 | 14.04.2022 09:00 - 12:15 | Online**
  Ant Colony Optimization | Foundations + Implementation
  `https://meet.goto.com/938184469`

  13:15 - 16:30 | Case study + Q&A

- **MOS-TINF19B | 19.04.2022 09:00 - 12:15 | Präsenz**
  Swarm Intelligence | Foundations + Boids
  Particle Swarm Optimization | Foundations + Implementation

- **MOS-TINF19A | 19.04.2022 13:15 - 16:30 | Präsenz**
  Swarm Intelligence | Foundations + Boids
  Particle Swarm Optimization | Foundations + Implementation

- **MGH-TINF19, MOS-TINF19A/B | 20.04.2022 09:00 - 12:15 | Online**
  Artificial Bee Colony Algorithm | Foundations + Implementation
  `https://meet.goto.com/296095741`

  13:15 - 16:30 | Case study + Q&A

- **MOS-TINF19A/B | 25.04.2022 09:00 - 12:15 | Online**
  Particle Swarm Optimization | Case study + Q&A
  `https://meet.goto.com/979352949`

- **MOS-TINF19A/B | 06.05.2021 09:00 - 12:15 | Online**
  Ant Colony Optimization | Case study + Q&A
  `https://meet.goto.com/180890741`

Prüfungsleistungen | Abgabetermin: 15.05.2022

Tutorial | 45 Punkte

- **Particle Swarm Optimization | 15 Punkte**

  Erstellung Word-/LibreOffice-Dokument mit prägnanter **Dokumentation in Deutsch** in eigenen Worten zu den Themenbereichen **[i]** Metapher, **[ii]** Strategie, **[iii]** Prozedur, **[iv]** Pseudocode, **[v]** Exploration und Exploitation, **[vi]** Entscheidungsregeln für Schwarmverhalten, **[vii]** Parameterabhängigkeiten, **[viii]** Zusammenspiel `gBest` und `pBest` mit Konfiguration `numberOfParticles = 20`, `inertia = 0.729844`, `cognitiveRatio = 1.496180`, `socialRatio = 1.496180`

- **Ant Colony System Algorithm | 15 Punkte**

  Erstellung Word-/LibreOffice-Dokument mit prägnanter **Dokumentation in Deutsch** in eigenen Worten zu den Themenbereichen **[i]** Metapher, **[ii]** Strategie, **[iii]** Prozedur, **[iv]** Pseudocode, **[v]** Exploration und Exploitation, **[vi]** Entscheidungsregeln für Schwarmverhalten, **[vii]** Parameterabhängigkeiten, **[viii]** Zusammenspiel `Ant` und `AntColony` mit Konfiguration `alpha = 2`, `beta = 2`, `evaporation = 0.05`, `q = 500`, `antFactor = 0.8`, `randomFactor = 0.01`

- **Artificial Bee Colony | 15 Punkte**

  Erstellung Word-/LibreOffice-Dokument mit prägnanter **Dokumentation in Deutsch** in eigenen Worten zu den Themenbereichen **[i]** Metapher, **[ii]** Strategie, **[iii]** Prozedur, **[iv]** Pseudocode, **[v]** Exploration und Exploitation, **[vi]** Entscheidungsregeln für Schwarmverhalten, **[vii]** Parameterabhängigkeiten, **[viii]** Zusammenspiel `Employed Bee`, `Onlooker Bee`, `Scout Bee`, Konfiguration `foodSource = 100`, `limit = 25`

Komplexaufgabe | 55 Punkte

- Die Bearbeitung dieser Aufgabenstellung erfolgt im Team mit zwei Studierenden, im Ausnahmefall ein Team mit 3 Studierenden.
- **Implementierung:** Java 17.0.2 | **keine Modellierung | keine JUnit-Tests**
- **Applikation 01** | 10 Punkte

  Für die Problemstellung **TSP** mit der **Dateninstanz a280** ist eine **Konsolen-Applikation für** die Suche mit **BruteForce** zu realisieren.

- **Applikation 02** | 30 Punkte

  Für die Problemstellung **TSP** mit der **Dateninstanz a280** ist eine **Konsolen-Applikation für** die **parallelisierte Optimierung** mit **wahlweise Particle Swarm Optimization**, **Ant Colony Optimization oder Artificial Bee Colony** zu realisieren.

  Die **Suche der Agenten** Particle, Ant oder Bee ist mit leistungsfähigen Technologien aus dem Paket `java.util.concurrent` zu **parallelisieren**.

  Die **Parameter** sind dahingehend zu **optimieren**, dass bezüglich dem bekannten Optimium 2579 eine **Lösungsqualität** von **mindestens 95%** erreicht wird.

  In einem **Logfile** ist das Schwarmverhalten nachvollziehbar zu protokollieren.

- **Applikation 03** | 15 Punkte

  Für die Problemstellung **TSP** mit der **Dateninstanz a280** ist eine **Konsolen-Applikation für** die **parallelisierte Suche einer bestmöglichen Parameterkonfiguration** für den in der Applikation 02 angewandten Algorithmus zu realisieren. Die bestmögliche Parameterkonfiguration wird in einer JSON-Datei gespeichert. Die Konsolen-Applikation zu Applikation 02 ist dahingehend zu erweitern, dass über Kommandozeile `-best [dateiname]` die JSON-Datei spezifiziert bzw. angewandt wird.

- **Zusatzaufgaben für Team mit 3 Studierenden**

  In einer **graphischen Oberfläche** - realisiert in **JavaFX**[1] - ist

  - das **Verhalten der Agenten bezüglich Exploration und Exploitation**
  - die **Kovergenz** in einem LineChart/XYChart (Schrittweite: 100)
  - je Agent die **Anzahl** in einem Histogramm von **Beiträgen** zur **besten Fitness**

  in Registern nachvollziehbar zu **visualisieren**.

  Durch Drücken der **Taste [F2]** wird das **LineChart/XYChart mit** dem **Histogramm in** eine **PDF-Datei** (Querformat) **exportiert**.

[1] https://openjfx.io/

2 Global Optimization Problem

A global optimization problem is defined as finding the parameter vector $\vec{x}$ that minimizes an objective function $f(\vec{x})$: `minimize` $f(\vec{x})$, $\vec{x} = (x_1, x_2, \ldots, x_{n-1}, x_n) \in \mathbb{R}^n$ which is constrained by the following inequalities and/or equalities $l_i \leq x_i \leq u_i$, $i = 1, \ldots, n$ `subject to:` [i] $g_j(\vec{x}) \leq 0$, for $j = 1, \ldots, p$ and [ii] $h_j(\vec{x}) = 0$, for $j = p+1, \ldots, q$.

3 Benchmark Functions

Rosenbrock

Dimension (d) : 2

$$f(x) = \sum_{i=1}^{d=1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{1}$$

Global minimum : $f(x^*) = 0$ at $x^* = (1, \ldots, 1)$

Drop-Wave

Dimension (d) : 2

$$f(x) = -1 + \frac{\cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2} \tag{2}$$

Global minimum : $f(x^*) = -1$ at $x^* = (0, 0)$

Cross-in-Tray

Dimension (d) : 2

$$f(x) = -0.0001\left(|\sin(x_1)\sin(x_2)\exp(|100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}|)| + 1\right)^{0.1} \tag{3}$$

Global minima : $f(x^*) = -2.06261$
. at $x^* = (1.3491, -1.3491), (1.3491, 13491), (-1.3491, 13491), (-1.3491, -13491)$

4 Swarm Intelligence

Swarm Intelligence is the discipline which researches on natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization.

The two basic search behaviors used Swarm Intelligence are exploration and exploitation. Exploration refers to searching the unexplored area of the feasible region while exploitation refers to the search of the neighborhood of a promising region.

5 Particle Swarm Optimization

Introduction

Particle Swarm Optimization (PSO) was first introduced by Dr. Russel C. Eberhardt und Dr. James Kennedy in 1995. As described by Eberhart und Kennedy, the PSO algorithm is an adaptive algorithm based on an social-psychological metaphor.

Particle Swarm has two primary operators [i] velocity update and [ii] position update.

During each generation each particle is accelerated toward the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from previous best position, and the distance from the global best position. The new velocity value is used to calculate the next position of the particle in the search space. This process is iterated, a set of number of times, or until a minimum error is achieved.

Inspiration

Particle Swarm Optimization is inspired by the social foraging behaviour of animals, e.g. flocking behaviour of birds.

Idea is similar to bird flocks searching for food.

Metaphor

Particles in the swarm fly through an environment following the fitter members of the swarm and biasing their movement.

- Particle Swarm Optimization imitates human or insects social behavior.
- Individuals interact with one another while learning from their own experience, and gradually move towards the goal.
- Each particle (or agent) evaluates the function to maximize at each point it visits in spaces.
- Each agent remembers the best value of the function found (*pbest*) and its co-ordinates.
- Secondly, each agent know the globally best position that one member of the flock had found, and its value (*gbest*).

Strategy

The goal of the algorithm is to have all the particles locale the optima in a multi-dimensional hyper-volume. This is achieved by assigning initially random positions to all particles in the space and small initial random velocities.

Procedure

Particle Swarm Optimization is comprised of a collection of particles that move around the search space influenced by their own best past location and the best past location of the whole swarm or a close neighbour.

Each iteration a particle's velocity is updating using

$$\begin{aligned} v_i(t+1) = v_i(t) + (c_1 \times \texttt{rand()} \times (p_i^{\text{best}} - p_i(t))) + \\ (c_2 \times \texttt{rand()} \times (p_{\text{gbest}} - p_i(t))) \end{aligned} \tag{4}$$

where $v_i(t+1)$ is the new velocity for the i^{th}, c_1 and c_2 are the weighting coefficients for the personal best and global best positions, $p_i(t)$ is the i^{th} particle's position at time t, p_i^{best} is the i^{th} particle's best known position, and p_{gbest} is the best position known to the swarm. The `rand()` function generate a uniformly random variable $\in [0,1]$.

A particle's position is updating using

$$p_i(t+1) = p_i(t) + v_i(t) \tag{5}$$

Pseudocode

initialise location and velocity of each particle

repeat

  for each particle **do**
    evaluate objective function for each particle
  end for

  for each particle **do**
    update best solution
  end for

  update best global solution

  for each particle **do**
    update velocity
    compute new locations of particles
  end for

until termination condition is satisfied

Algorithm

Algorithm 1 Particle Swarm Optimization

Input: problemSize, populationSize

Output: $p_{g.best}$

```
population ← 0
p_g.best ← 0
for i = 1 to populationSize do
    p_velocity ← randomVelocity()
    p_position ← randomPosition(populationSize)
    p_cost ← cost(p_position)
    p_p.best ← p_position
    if p_cost ≤ p_g.best then
        p_g.best ← p_p.best
    end if
end for
while ¬ stopCondition() do
    for all p ∈ population do
        p_velocity ← updateVelocity(p_velocity, p_g.best, p_p.best)
        p_position ← updatePosition(p_position, p_velocity)
        p_cost ← cost(p_position)
        if p_cost ≤ p_p.best then
            p_best ← p_position
            if p_cost ≤ p_best then
                p_g.best ← p_p.best
            end if
        end if
    end for
end while
return p_g.best
```

Parameter | Heuristics

- **Swarm size**, i.e. the number of particles in the swarm: The more particles in the swarm, the larger the initial diversity of the swarm. Large swarm allows larrger parts of the search space to be covered per iteration. More particles increase the per iteration computational complexity, and the search space degrades to a parallel random search.

  It has been shown in a number of empirical studies that PSO find optimal solutions with small swarm sources of 10 to 30 particles. While empirical studies give a general heuristic of $n \in [10, 30]$, the optimal swarm size is problem-dependent.

- **Neighbourhood size**: The neighbourhood size defines the extent of social interaction within the swarm. The smaller the neighbourhoods, the less interaction occurs. While smaller neighbourhoods are slower in convergence, they have more reliable convergence to optimal solutions. Smaller neighbourhood sizes are less susceptible to local minima.

  To capitalize on the advantages of small and large neighbourhood sizes, start the search with small neighbourhoods and increase the neighbourhood size proportionally to the increase of number of iterations. This approach ensures an initial high diversity with faster convergence as the particles move towards a promising search area.

- **Number of iterations**: The number of iterations to reach a solution with good quality is problem-dependent. Too few iterations may terminate the search prematurely. A too large number of iterations has the consequence of unnecessary added computational complexity (provided that the number of iterations is the only stopping condition).