List of exercises 02
Metaheuristics

Augusto Mathias Adams[*]
[*]augusto.adams@ufpr.br

Abstract—**This document is submitted as a partial requirement for evaluation of the discipline of *Evolutionary Computation and Inteligence of Swarms(TE-943)* of the Electrical Engineering Course, with emphasis in Embedded Systems, of the Electrical Engineering Department at the Federal University of Paraná.**

Index Terms—**Optimization Problem; Linear Programming; Non-linear Programming;**

I. Question 01

Given the design vector: $\vec{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots \mathbf{x}_n\}^T$***, match the following terms and descriptions:***

II. What are the differences between heuristics, metaheuristics,and hyperheuristics?

Heuristics are simple, practical methods for finding approximate solutions to problems that are not guaranteed to be optimal. They are based on a set of rules or strategies that are usually derived from experience and expert knowledge.

Metaheuristics are algorithms that use heuristics to solve complex optimization problems. They are more sophisticated than heuristics, as they adapt and change the heuristics they use based on the problem and the results of previous iterations.

Hyperheuristics are a level above metaheuristics, as they use higher-level strategies to choose and control the metaheuristics used to solve a problem. Instead of using a fixed set of heuristics, hyperheuristics dynamically select and adjust the heuristics they use based on the problem, performance, and other factors.

In summary:

- Heuristics are simple, practical methods for finding approximate solutions.
- Metaheuristics are algorithms that use heuristics to solve complex problems.
- Hyperheuristics are algorithms that use higher-level strategies to control and choose the metaheuristics used to solve a problem.

III. Question 03:

IV. Question 04:

A. What is the difference between exploration and exploitation?

Exploration and exploitation are terms used in decision making and refer to two different approaches in achieving a goal.

Exploration refers to seeking out new and diverse options and trying new things, often at the risk of not immediately achieving the goal. The goal of exploration is to gather information and increase the available options to make a more informed decision in the future.

Exploitation, on the other hand, refers to using the information already gained to make the most of the current options in order to achieve the goal as efficiently as possible. The goal of exploitation is to make the best use of known information and resources to maximize rewards or benefits.

In summary, exploration involves seeking new information, while exploitation involves using existing information.

B. What is the difference between intensification and diversification?

Intensification and diversification are two important concepts in metaheuristics and hyperheuristics.

Intensification refers to the process of focusing search efforts on a promising region of the search space. It aims to improve the quality of the solutions in that region by applying more intensive search techniques, such as local search, hill climbing, or simulated annealing. Intensification is used to refine and improve solutions that are already promising, to increase the chance of finding an optimal solution.

Diversification, on the other hand, refers to the process of expanding the search efforts to different regions of the search space. It aims to explore new and potentially promising regions of the search space that have not yet been explored. Diversification is used to avoid getting stuck in a local optimum, to escape from an unfavorable situation, or to find new and better solutions.

In summary:

- Intensification focuses search efforts on a promising region to refine and improve solutions.
- Diversification expands search efforts to different regions to explore new and potentially better solutions.

C. What is the difference between global search and local search?

Global search and local search are two types of search strategies used in optimization algorithms.

Global search refers to a type of search strategy that explores the entire search space to find the best solution. It aims to find a globally optimal solution by considering all possible solutions and choosing the best one. Global search algorithms are typically used to solve problems where the solutions are not easily predictable and the search space is large. Examples of global search algorithms include genetic algorithms, particle swarm optimization, and simulated annealing.

Local search, on the other hand, refers to a type of search strategy that focuses on refining solutions that are already promising. It aims to find a locally optimal solution by making small, incremental changes to the current solution. Local search algorithms are typically used to solve problems where the solutions are predictable and the search space is small. Examples of local search algorithms include hill climbing, gradient descent, and tabu search.

In summary:

- Global search explores the entire search space to find the best solution.
- Local search focuses on refining solutions that are already promising.

V. How the exploration and exploitation strategies are designed in a standard particle swarm optimization approach?

In Particle Swarm Optimization (PSO), the exploration and exploitation strategies are typically designed through the use of velocity and position updates for each particle in the swarm. The velocity update for each particle is a combination of the particle's current velocity, the particle's "personal best" position, the "global best" position found by the entire swarm, and a random factor. The position update is simply the current velocity added to the particle's current position. By adjusting the relative weights of the various components in the velocity update, the balance between exploration and exploitation can be controlled. In general, a higher weight for the global best position will lead to more exploitation, while a higher weight for the random factor will lead to more exploration.

VI. We want to a binary GA (Genetic Algorithm) to find x to a resolution of 0.1 to minimize the two-dimensional Rastrigin function ($n = 2$) on the domain $[-5,5]$.

A. How many genes do we need for each chromosome?

The number of genes needed to solve the Rastrigin function in two dimensions with a genetic algorithm depends on a number of factors such as population size, mutation rate, selection method, and stopping criteria. Typically, more genes lead to more complex representations of potential solutions and may result in better solutions, but may also increase computation time. A good starting point would be to use a chromosome length equal to the number of dimensions in the problem (2 in this case), and adjust as needed based on experimentation.

B. How many bits do we need in each gene?

To solve the Rastrigin function in two dimensions with a resolution of 0.1 using a genetic algorithm, you need to use enough bits in each gene to represent the desired range and precision of the solution. The Rastrigin function is defined over the range of -5.0 to 5.0 for both dimensions, so you need to encode a value in each gene that can represent this range.

To encode a value in a binary representation with a resolution of 0.1, you need to use a number of bits that can represent the range (-5.12, 5.12) with increments of 0.1. You can estimate the number of bits required by taking the base-2 logarithm of the range divided by the precision, and then rounding up to the nearest integer.

In this case, the number of bits required is approximately log2((5.0-(-5.0))/0.1) = log2(10.0/0.1) = log2(100) = 6.64, so you would need to use 7 bits in each gene.

C. title

VII. How could you change the differential evolution algorithm to be non-elitist?

A non-elitist Differential Evolution (DE) algorithm would differ from the standard elitist DE algorithm in the way the best candidate solution is selected. In the elitist DE, the best candidate solution is always preserved for the next generation. However, in a non-elitist DE, the best candidate solution may be replaced by a worse solution if it leads to a better overall result.

To implement a non-elitist DE algorithm, you could change the selection mechanism so that the algorithm doesn't always preserve the best candidate. This can be done by introducing a random component to the selection process, so that the best candidate is not always chosen.

VIII. What are the differences between standard (classical) evolutionary programming and evolution strategy when applied to continuous optimization?

Evolutionary Programming (EP) and *Evolution Strategy* (ES) are two optimization algorithms used to solve problems in continuous search spaces. The main differences between EP and ES are:

- ***Selection method:*** In EP, selection is done based on the fitness of the individuals. In ES, selection is done based on the performance of the parents and their offspring, rather than only considering the fitness of the parents.
- ***Mutation:*** In EP, mutation is performed on the entire chromosome at once. In ES, mutation is performed by adding random noise to the individual's parameters.
- ***Variation operator:*** In EP, the variation operator is typically a combination of mutation and crossover. In ES, mutation is the only variation operator used.
- ***Reproduction:*** In EP, reproduction is performed by creating a new generation of individuals. In ES, the new generation is created by adding random noise to the parents and their offspring.
- ***Parent-offspring relationship:*** In EP, the relationship between the parent and offspring is not explicitly defined. In ES, the parent-offspring relationship is explicit and the offspring are created by adding noise to the parents.

EP is generally considered to be a more "explicit" optimization algorithm, while ES is considered to be a more "implicit" optimization algorithm. Both algorithms can be effective for solving continuous optimization problems, but the choice between the two depends on the specific requirements of the problem being solved.

IX. Present step-by-step to the approach named harmony search algorithm (Geem et al., 2001) applied to a general continuous optimization problem?

- ***Initialization:*** Initialize the variables required for the HSA such as the harmony memory size, harmony memory considering rate, pitch adjusting rate, and the stopping criteria.
- ***Harmony Memory Initialization:*** Initialize the harmony memory with random solutions.
- ***Fitness Evaluation:*** Evaluate the fitness of the solutions in the harmony memory.
- ***Improvisation:*** Improvise the solutions in the harmony memory by adjusting the pitch or generating new solutions using randomization and exploitation.
- ***Harmony Memory Update:*** Update the harmony memory with the best solutions.
- ***Stopping Criteria:*** Check if the stopping criteria have been met. If the criteria have been met, the algorithm terminates, otherwise, return to step 3.
- ***Return the best solution:*** Return the solution with the highest fitness as the optimal solution.