Digital Systems Project

Search algorithm to be embedded with a reinforcement learning algorithm to solve optimization problems.

Tom South – 18011748

University of the West of England, Bristol

UFCFXK-30-3

# Abstract

This report proposes a swarm intelligence algorithm that will work with multiple binary operators which will run through a reinforcement learning driven selection scheme and the algorithm will be able to solve binary problems such as the 0-1 multidimensional knapsack problem. “Swarm intelligence concerns the collective, emerging behavior of multiple, interacting agents that follow some simple rules.” Swarm intelligence acts like collective intelligence where agents show organization in their behavior. (Hassanien and Emary, 2016)

A program will be designed, implemented and tested using different information collected during research, using ideas found in literature read and even taking ideas of areas not been fully identified. The purpose of this program is to use a reinforcement driven selection to automatically select the swarm intelligence algorithms operators, such as mutation, to solve binary problems.

# Acknowledgements

# Table of Contents

[**Abstract** 2](#_Toc57907475)

[Acknowledgements 3](#_Toc57907476)

[Table of Contents 4](#_Toc57907477)

[1. Introduction 5](#_Toc57907478)

[2. Literature Review 6](#_Toc57907479)

[2.1 Previous Projects 8](#_Toc57907480)

[3. Requirements 9](#_Toc57907481)

[4. Methodology 10](#_Toc57907482)

[4.1 Planning 10](#_Toc57907483)

[4.1.1 Sprint 1 10](#_Toc57907484)

[4.1.2 Sprint 2 11](#_Toc57907485)

[4.1.3 Sprint 3 11](#_Toc57907486)

[References 12](#_Toc57907487)

# Introduction

Optimization means finding the best solution among many feasible solutions that are available to us. You base a function off its performance, and the aim is for the function to be maximized/minimized (Arora, 2015). Swarm intelligence-based algorithms are from an area called bio-inspired algorithms, e.g. genetic algorithms are also from this area, as they are natured inspired, the particle swarm optimization I will be using is natured inspired, also are others such as the ant colony optimization (Hassanien and Emary, 2016). Reinforcement learning agents are among the most advanced and very capable of demonstrating high level of intelligence and rational behavior (Sewak, 2019). Reinforcement learning interacts with its environment and due to the environment it is in, can change many different states, it can use a point system as a reward system, based on how it changes the environment it is in and the objective it wants to complete. Sewak (2019) states that reinforcement learning led to the development of many advanced AI agents to perform very complex tasks, sometimes even challenging human performance at specific tasks.

For this project, research will be conducted on how swarm intelligence, particularly particle swarm optimization can be used to solve such binary problems, where the 0-1 multidimensional knapsack problem will be intensively researched as the main problem to be solved. Reinforcement learning will be researched and developed as the selection scheme, as reinforcement learning rewards agent’s dependent on its actions therefore creating a learning process over time. The aim is for the reinforcement learning algorithm to teach the optimization algorithm how to solve this problem. (Ravichandran, 2018).

One of the problems being tested is the 0-1 multidimensional knapsack problem, this problem arises in different scenarios, Boyer, Elkihel and El Baz (2009) states that these problems are found in cargo loading, cutting stock problems and processor allocation. This problem will be intensively researched and will be the benchmark for testing my program as it is a binary problem.

# Literature Review

(Detail Particle Swarm Optimization and Reinforcement learning properly.)

Nandy and Biswas (2018) state how reinforcement learning starts with an intelligence program, known as agents, and when they react with environments, there are rewards and punishments, environments can be known or unknown to the agents. These agents take specific actions to move to continuous states so that they reach their goal by maximizing the rewards given. Figure 1 shows the flow of reinforcement learning as stated by Nandy and Biswas (2018).

Diagram

Description automatically generated

*Figure 1.0: Reinforcement Learning Flow.*

Sewak (2019) details how the reward is a function of both the action and the state, not just the action by itself, therefore the same action could (and ideally should) receive a different reward under different states, which under real-life terminology explains how using different actions in different aspects and areas of life, give different rewards and consequences. This details how impressive the approach of reinforcement learning is compared to other learning approaches, as it considers the environment in which actions are taken.

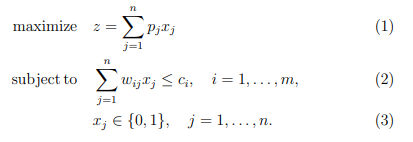
(Explain particle swarm optimization.)

Haddar, Kehmakhem, Rhimi and Chabchoub (2016) state that the particle swarm optimization algorithm is a heuristic global optimization method originally introduced by Kennedy and Eberhart in 1995 and that the algorithm exploits the concept that knowledge is needed for the search of an optimal solution and can be based on observed social behavior. Giftson Samuel and Christober Asir Rajan (2015) follow the previous statement by explaining how it is inspired from the collective behavior stated above in swarms of social insects. It explains how a swarm is made up of many particles and each particle represents an individual. Each particle is dedicated their own position and flight velocity, which during the optimization process becomes adjusted.

They continue to explain how the initial particles and velocities get initialized through random selection, then the velocities get updated by updating the fitness of each particle in the swarm, then the position of each particle is updated. This is something I want to explore creating in my own project to solve my multi-dimensional knapsack problem, I will look for research in previous work of using PSO and RL in binary problems and look for areas which I can see a gap and can improve. (Maybe change this.)

Researching MKP in general.

Punchinger et al. (2010) researched the structure and algorithms of the MKP and as shown in Figure 1.1 that the MKP can be defined as:



*Figure 1.1: Multidimensional Knapsack Problem (MKP)*

Puchinger et al. (2010) follow up this formula by explaining “A set of *n* items with profits *pj* > 0 and *m* resources with capacities *ci* > 0 are given. Each item j consumes an amount *wij* ≥ 0 from each resource *i*. The 0-1 decision variables *xj* indicate with items are selected.” The goal of the problem is to choose a subset of items with maximum total profit. But selected items must not exceed some resource capacities, which are expressed by the knapsack constraints in (2) of Figure 1.1. I will continue to research the MKP by looking at areas where the MKP has been solved using different forms of optimization.

## 2.1 Previous Projects

(Information about past uses of reinforcement learning and particle swarm optimization)

Reinforcement learning can be used in many different aspects, Ciaburro (2018) details different examples where he has used reinforcement learning in different use cases with different algorithms. The main one researched is the Balancing Cart Pole as it uses OpenAI Gym framework, which will be the framework I use, it is also a traditional problem and a well-known problem. Ciaburro (2018) explains how the objective is to stabilize the pole, without it falling to the ground, the agent will choose to move the cart left or right, and it will receive a reward every time the pole is balanced. The problem is a simple binary classification problem as it contains four inputs (cart position, cart velocity, pole angle, pole velocity at tip) and one single binary output.

(Researching MKP projects)

Azad, Rocha and Fernandes (2014) created an artificial fish swarm algorithm for the 0-1 MKP where the artificial fish uses the entity of a real fish, where the school is the population of fish, therefore the environment that the fish moves in, searching for the best solution, is the search space of the MKP problem. The fish in the swarm go through a chasing and swarming behavior, which they show through an algorithm to compute the central point which is the point closest to all other points, the swarm then “searches” and behaves randomly based on the “visual scope” They have a penalty function method to handle the constraints, very similar to what you see from the research in reinforcement learning, where it is penalized/rewarded dependent on the state of the environment. The program then goes through improving the feasible points given, which then they select a new population. Researching this journal containing this project was very interesting as it had a similar sense to what I want to do, just with particle swarm optimization, even though how it is handled is different, the overall basic outline is shown on how to handle this problem using swarm algorithms.

# Requirements

There are 6 functional requirements and 6 non-functional requirements listed below which are the outline for what I expect my program to do, and how I expect data to be outputted/shown. These requirements either use “must”, “should” and “could” to prioritize what requirements are necessary and what requirements are not but would be a good addition to the program.

|  |  |
| --- | --- |
| Requirement | Description |
| F1 | The program must be able to solve the 0-1 multidimensional knapsack problem. |
| F2 | The program must have a working reinforcement learning algorithm that rewards/penalizes the agent due to the state of the environment it is in. |
| F3 | The program must have working operators within the particle swarm optimization, so it can optimize the problem and get a solution. |
| F4 | The program must be able to show that optimization has happened and that it is solving the 0-1 multidimensional knapsack problem in stages. |
| F5 | The program should allow for the reinforcement learning algorithm to choose which operators should be run in the particle swarm algorithm. |
| F6 | The program should allow for the agent to work in all states of the environment allowing for a penalize/reward situation. |
| NF1 | The program should have a clear structure on how it is solving the 0-1 MKP. |
| NF2 | The program should retrieve the data quickly. |
| NF3 | The program should be able to run on any OS. |
| NF4 | The program should present how it is solving the data by every time there is a |
| NF5 | The program should show how long it took to solve the problem. |
| NF6 | The program could write its information into csv files/text files to show in different runs how they compare in performance. |

*Table 1: Requirements.*

# Methodology

I have decided to use the Agile method while implementing my project, as this allows for me to receive feedback especially if I come across problems during the implementation period, which I am certain will arise. This feedback will allow me to make changes in the correct areas and raise new ideas on how I can implement my project.

I have decided to use Sprints to spread out how I implement my project, which I will show below in my planning, I decided to use sprints because of the experience I have in using them, it also allows me to work to certain deadlines where I can get feedback after these deadlines and can go back to finalize areas I have worked on. Overall, I have a good understanding of using sprints and I believe this to be the optimal way for me to do my project.

After each sprint, I will aim to get feedback on what I have produced and attempt to make slight changes in accord to the feedback that I have been given, and finally “clean up” any code that I have created.

## 4.1 Planning

I have used TeamGantt to create my Gantt chart due to the simplicity of the design, it also allows you to change it very easily as you go through the project, using percentages to show the completion so far of the project. I would change this during the project to accommodate any changes I have done in the order of my implementation / testing or any other action. Figure 1 is my complete Gantt Chart once development has been completed.

(Complete gantt chart.) – At finish.

*Figure 1: Gantt Chart.*

### Sprint 1

The first Sprint is designed to develop a particle swarm optimization (PSO) algorithm, which is the base of the program, this will allow me to then set up a reinforcement learning algorithm to implement the reward system dependent on the particle swarm optimization’s state. During this period, I will be creating operators for the optimization, such as swarming, selection, mutation. This will then be tested on test cases to make sure the algorithm can optimize problems and at the end of the sprint, there will be a time for feedback and changes.

### Sprint 2

The second sprint is implementing the reinforcement learning selection scheme to reward/penalize the particle swarm optimization and choose the operators in the PSO to improve the performance. This process could be the most lengthy of the sprints as a brand new algorithm needs to be created and the PSO might have to be changed to work with the RL algorithm, extensive testing will also be needed so I am able to move onto the next sprint.

### Sprint 3

The final sprint is to make sure my finished program containing the PSO and RL algorithms fully optimize the 0-1 MKP. This will be done through extensive testing and might require changes therefore a whole sprint will be required to make sure the full algorithm works correctly, the final part of the sprint will be designated to bug fixing if there are any and final changes and testing and creating test cases.

# References

Hassanien, A.E and Emary, E. (2016), Swarm Intelligence: Principles, Advances and Applications. [online], CRC Press. [Accessed 19 November 2020]

Ravichandiran, S. (2018), Reinforcement Learning with Python [online], Packt Publishing. [Accessed 19 November 2020]

Arora, R.K. (2015­), Optimization: Algorithms and Applications[online], CRC Press. [Accessed 19 November 2020]

Sewak, M. (2019), Deep Reinforcement Learning: Frontiers of Artificial Intelligence [online], Springer. [Accessed 19 November 2020]

Boyer, V., Elkihel, M. and El Baz, D. (2009) Heuristics For the 0-1 Multidimensional Knapsack Problem. European Journey of Operational Research [online]. 199 (3) [Accessed 19 November 2020].

Ciaburro, G. (2018) Keras Reinforcement Learning Projects [online], Packt Publishing. [Accessed 19 November 2020].

Nandy, A. and Biswas, M. (2017) Reinforcement Learning [online]. : Apress. [Accessed 19 November 2020].

Haddar, B., Khemakhem, M., Rhimi, H. and Chabchoub, H. (2016) A Quantum Particle Swarm Optimization For the 0-1 Generalized Knapsack Sharing Problem. Natural Computing [online]. 15 (1) [Accessed 22 November 2020].

Giftson Samuel, G. and Christober Asir Rajan, C. (2015) Hybrid: Particle Swarm Optimization-genetic Algorithm and Particle Swarm Optimization-shuffled Frog Leaping Algorithm For Long-term Generator Maintenance Scheduling. International Journal of Electrical Power & Energy Systems [online]. 65, pp. 432--442. [Accessed 22 November 2020].

Punchinger, J., Raidl, G.R. and Pferschy, U. (2010) The Multidimensional Knapsack Problem: Structure and Algorithms. Informs Journal on Computing [online]. 22 (2), pp. 250-265. [Accessed 22 November 2020].

Azad, M.A.K., Rocha, A.M.A.C. and Fernandes, E.M.G.P, (2014) Improved Binary Artificial Fish Swarm Algorithm For the 0–1 Multidimensional Knapsack Problems. Swarm and Evolutionary Computation [online]. 14 [Accessed 03 December 2020].