MULTIMEME MEMETIC MIGRATING BIRDS OPTIMIZATION FOR QUADRATIC ASSIGNMENT PROBLEM

by

Emir Said Haliloğlu
Mustafa Yanar
Hasan Fatih Başar

CSE4197 / CSE4198 Engineering Project report submitted to Faculty of Engineering in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

Supervised by:
Prof. Dr. Ali Fuat ALKAYA

Marmara University, Faculty of Engineering
Computer Engineering Department

2024

MULTIMEME MEMETIC MIGRATING BIRDS OPTIMIZATION FOR QUADRATIC ASSIGNMENT PROBLEM

by

Emir Said Haliloğlu

Mustafa Yanar

Hasan Fatih Başar

CSE4197 / CSE4198 Engineering Project report submitted to Faculty of Engineering in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

Supervised by:

Prof. Dr. Ali Fuat ALKAYA

Sign

Marmara University, Faculty of Engineering

Computer Engineering Department

2024

ABSTRACT

Quadratic Assignment Problem (QAP) is an NP-Hard optimization problem with the model such that there are a set of n facilities and a set of n locations of these facilities. Between the facilities the distances are known, and the problem is to figure out the best permutation for placing n facilities in n locations. The MBO (Migrating Birds Optimization) algorithm is a type of neighborhood search technique that involves a set of initial solutions represented as birds in a V formation [1]. This project's aim is to improve Migrating Bird Optimization (MBO) for solving the QAP. The approach is to achieve this by hybridizing MBO with the Multimeme Memetic algorithms by effectively defining the necessary operators designed for QAP. Although these two algorithms use different methodologies, they have the same purpose. Within the scope of the project, we have compared the results of some certain data sets named QAPLIB which are also used by MBO to compare benchmark results. As conclusion of the project, we have successfully improved MBO for some certain data sets.

ACKNOWLEDGEMENTS

I would also like to extend my thanks and gratitude to my advisor for his guidance, encouragement, and patience. I express gratitude to all the Department faculty members for their help and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Quadratic Assignment Problem (QAP) belongs to a class of optimization problems known as NP-hard, characterized by their inherent computational difficulty and lack of known efficient algorithms for finding optimal solutions. Sharing its complexity with prominent problems like the Traveling Salesman Problem (TSP), the QAP presents a significant challenge in the realm of NP-hard combinatorial optimization.

1.1 Problem Description and Motivation

Quadratic Assignment Problem (QAP) is a combinatorial optimization problem that involves assigning a set of facilities to a set of locations, with the objective of optimizing a quadratic cost function. Each pair of facilities incurs a cost associated with their interaction, and each pair of locations includes a cost based on their separation. The goal is to determine the optimal assignment of facilities to locations that minimizes the total cost.

Large-scale combinatorial optimization issues are typically unsolvable; therefore, one must settle with less-than-ideal solutions. Heuristic algorithms, which are often grouped as constructive and improvement algorithms, find near optimal solutions.

Main motivation was to develop a metaheuristic solution for QAP to increase the quality of the solution by minimizing the cost. Since QAP is one of main problems of operations research and real world, finding a better algorithm for QAP would contribute a good effect to the real world. Our contributions can reflect in a good way and can be a good source for other researchers who are working on these kind topics. We hybridized these algorithms that we mentioned above, and the algorithms will be used to assist by some of their properties and operators while we are trying to solve the QAP.

1.2 Main Goal and Objectives of the Project

We aimed to enhancing the quality of the solution by combining two metaheuristic algorithms, namely the Migrating Birds Algorithm and the Multimeme Memetic Algorithm, to effectively address the QAP. Although these two algorithms use different methodologies, they have the same purpose. We will

be comparing our combined algorithm with other approaches to solving the quadratic assignment problem within a given time.

- **Objective #1 – Analyzing Algorithms:** Analyzing both Migration Birds Optimization Algorithm and Multimeme Memetic Algorithms. Working for examining how they work, their effectiveness, and where they can be used for optimization.
- **Objective #2 – Detecting Spots for Implementation**: Identifying suitable components for combination within both algorithms. For those components, detect the spots for combined version of algorithms and identify the convenient parts for operators which are assisting us to improve solution.
- **Objective #3 – Implement Operators and Apply Tests with QAPLIB:** Implementing the operators is one of the most efficient parts for this project. Since we use various operators, they can enlarge algorithm's perspective to both exploration and exploitation. After implementing operators, we are going to test algorithm with QAPLIB.
- **Objective #4 – Evaluating the Result and Identify the Specialty of Algorithm:** Adaptive Multimeme Memetic Migrating Birds Optimization algorithm has been designed to get better result on QAP. It aims to get balance on exploration and exploitation dilemma. We have compared the results of both adaptive and non-adaptive MMMBO and other algorithms to evaluate the performance of algorithm

2. DEFINITION OF THE PROJECT

2.1 Scope of the Project

In our project, we implemented an improved genetic algorithm approach which is combined migrating birds' optimization to solve quadratic assignment problem. We will observe the methodologies of Multimeme memetic algorithm on QAP and implement on MBO. Memetic algorithm uses operators for the different steps of it to decide for pick behavior. Therefore, we are going to find the best operator options based on QAP. Some key points we'll track is described below:

- We observed the performance of crossover operators, mutation operators, local search operators and ruin-recreate operators on QAP.
- When our Multimeme memetic algorithm model was designed, we integrated the MBO into our model in the right way and developed our model. The MMA was a co-evolutionary algorithm, but MBO was a local search optimization, so we added another step in the process of producing the next generation.
- For the test data, there is an open-source library named QAPLIB. Quadratic assignment problem library is used to test our results and to compare them with other approaches.

2.2 Success Factors

For our project, we can talk about various key factors as follows:

- **Innovative Approach:** This project can differentiate itself by putting forward new approaches for hybridization or by altering current algorithms. Innovative methods and concepts for merging algorithms have the potential to significantly improve the quality of the solutions.
- **Understandable Implementation:** One of the success factors is writing clean and understandable code. For the legacy project and other implementations of metaheuristic approaches, we had to take care of keep the understandability of the code. Since we hybridize different methodologies, it can confuse people's minds and if we keep the understandability level high, it is counted as a success.
- **Effective Methodology:** It is important to have a clear, methodical approach for breaking down, improving, and analyzing the algorithms. The validity and

reliability of the results are enhanced by precise testing procedures, suitable performance measures, and well-defined experimental designs.

- **Experimental Design:** Our project needed to be tested by datasets which are recognized globally by other researchers. For the test and benchmark results, we needed to design our experiments in a well-shaped format. For this purpose, we used QAPLIB which is mentioned in 3.1.4.
- **Algorithm Optimization**: In our project, since we want to improve the solution, through the development of new methods, parameter adjustments, or the introduction of innovative ideas, the project seeks to produce optimum results that beat current standards. The achievement of ideal solutions and the demonstration of significant performance improvements highlight the influence and success of the research efforts. This would be one of the biggest success factors.

2.3 Professional Considerations

2.3.1 Methodological Considerations / Engineering Standards

In the development of the project, to keep track of versions and control the source code Git is used. Legacy code for MBO was written in Java, we implemented our contributions in Java as well. To watch and compare data, we made an application to make lists in Excel file. Excel's visualization tools are used to create graphs and tables to compare the results.

Flow charts are used to show the algorithm logic and operators from different kinds of algorithms.

2.3.2 Realistic Constraints

- **Economic:** Equipment to run the algorithm to use the project is the only cost for our project. Since everything is written in Java, no framework or paid tools are needed. If the output is tried in real life, there might be some additional costs.
- **Environmental:** Our project can be applied to different kinds of instances in the environment. Processes in our project can lead to reduced energy consumption and lower environmental impact.

Since it is a solution for optimization, it doesn't affect the environmental pollution in a bad way.

- **Ethical:** Our project doesn't cause any ethical problems.
- **Health and Safety:** If the user watches rules of using a computer, there is no cause to danger for health and safety.
- **Sustainability:** Since our project can lead to reduced energy consumption, it keeps its sustainability.
- **Social:** Our project does not cause any social discrimination.
- **Legal Considerations:** There is no legal consideration since we used Java and licensed Excel.

2.4 Literature Survey

We had a close look at what other researchers have studied so far. This helped us understand the background and build on what's already known as we work on solving the Quadratic Assignment Problem using a mix of different problem-solving methods.

Mathew Harris et al. [2] proposed a memetic algorithm with parallel local search using a population with three structure [6-9] and local search operator based on Tabu Search. It is an extension for "Evaluating memory schemas in a Memetic Algorithm for the QAP" [3]. The Multimeme or memetic algorithm for quadratic assignment problem utilizes conventional methods such as local search, selection, and crossover for reproduction, and implements a 7-restart mechanism when population diversity is low. It employs a ternary tree population structure successfully used by [6-9], where each node stores a set of solutions. The project also shows and experiments with various Tabu Search variants and implements the algorithm efficiently through parallelization across multiple CPU cores. The proposed algorithm is compared robustly on QAP instances from QAPLIB [10]. A comparison is made with a MA lacking a structured population to benchmark the performance of the tree structure. The proposed algorithm is also compared to various state-of-the-art methods in the literature, and conclusions are drawn based on the results obtained on its performance.

Tabu search, proposed in 1986 [4], is a highly effective metaheuristic method that has been widely applied to solve various optimization problems. It is based on the idea of using human memory as a metaphor to track solution

trajectories, and its efficient search strategies have been shown to produce excellent results for difficult optimization problems [5]. The main design factors of the generic Tabu search algorithm include the number of iterations, the size of the tabu list, and the size of the neighborhood. Here we have a pseudocode to show how to obtain tabu search. Pseudocode for Tabu Search implementation is provided below.

Simulated Annealing (SA) was developed as a means of modeling optimization problems using the analogy of annealing solids [11]. This process involves heating solid particles and allowing them to randomly re-arrange themselves before cooling them down according to a specific schedule until they reach a low-energy state. SA uses this metaphor to represent the optimization problem, with the solid particles representing potential solutions and the energy level indicating the fitness or objective function value. Various studies have explored the applicability of SA in combinatorial and continuous optimization contexts. However, only a few have examined the impact of algorithm design and parameter tuning on combinatorial optimization, particularly for large-scale problem instances [12]. To address this gap, this study employs a general mathematical model of SA that utilizes random key solution representation and an iterative parameter setting process to evaluate the performance of each design. In the general SA model, temperature is a crucial factor that determines the quality of the solution. The particles are stabilized in thermal equilibrium at each temperature, and the energy is described using the Boltzmann distribution.

Genetic Algorithms (GAs) are widely employed to find effective solutions for optimization and search problems. These algorithms belong to a broader category called evolutionary algorithms (EAs), which employ techniques inspired by natural evolution to generate solutions for optimization problems. In a genetic algorithm, a population of strings, referred to as chromosomes or the genotype of the genome, represents potential solutions, also known as individuals or creatures. The goal is for this population to evolve towards better solutions. Traditionally, solutions are represented as binary strings of 0s and 1s, although alternative encodings are possible. The algorithm begins with a population of randomly generated individuals and progresses through generations. In each generation, the fitness of everyone in the population is assessed. Based on their fitness, multiple individuals are probabilistically selected from the current

population and subjected to modifications, including recombination and occasional random mutations, to form a new population. This new population is then used in the subsequent iteration of the algorithm. The algorithm typically concludes either when a maximum number of generations has been reached or when a satisfactory fitness level has been attained for the population. In the case of reaching the maximum number of generations, it is uncertain whether a satisfactory solution has been obtained or not. [13].

3. SYSTEM DESIGN AND SOFTWARE ARCHITECTURE

3.1 System Design

In this section, we will present the system design. Our methodology is to improve the MBO by MMA algorithm to get more efficient results on QAP. We first mention the problem definition theoretically.

3.1.1 System Model

QAP is an assignment problem which contains n point and $n\ x\ n$ weight matrix. The aim in the problem is to find an assignment in which all facilities are assigned different locations. To optimize the problem, total cost should be the lowest in the possible solution space. Let us describe with these steps:

1. **Input Data:** n location and $n\ x\ n$ weight matrix is given. Weight matrix defines that the cost will be occurred when two points are placed.
2. **Assignment Matrix:** P is a matrix which $n\ x\ n$ size and shows the assignment. Value of Pij would be 1 in case of point i assigned point j otherwise 0. A partial equation (3.1)

$$P_{ij} = \begin{cases} 1, & if\ point\ i\ is\ assigned\ to\ point\ j \\ 0, & otherwise \end{cases} \quad \text{(Equation 3.1)}$$

3. **Weight Cost:** It is calculated by using Pij. Weight cost express that total cost of the situations that placed two points (i and j). This cost is calculated by multiplying weight matrix and assignment matrix.
4. **Finding An Optimal Solution**: The goal at solving the problem is to optimize assignment problem and find a solution which reduces the total weight cost to the lowest. This is a non-linear optimization problem, and it is solved by using various optimization technics or metaheuristic algorithms. An equation for that (3).

$$Total\ Weight\ Cost\ f(P) = \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} \cdot P_{ij} \cdot \sum_{k=1}^{n}\sum_{l=1}^{n} P_{kl} \cdot (1 - \delta_{ik}) \cdot (1 - \delta_{jl})$$

(Equation 3.2)

Here, δ_{ik} is the Kronecker delta function, which equals 1 if $i = k$ and 0 otherwise. (3.2)

3.1.2 Flowchart and/or Pseudo Code for Proposed Algorithms

Migrating Bird Optimization is a metaheuristic approach inspired by migration of birds to solve complex problem. It is a neighborhood search technic. In the algorithm, the V formation of bird is used. Each bird represents a possible solution for the problem. Initially random solutions for initial flock are selected for birds in the V formation. Then iteratively from the leader to tail, each bird (solution) tries to improve the solution of that position in the formation by its neighbor solutions (for the implementation of QAP, a neighbor solution is obtained by pairwise exchange (Figure3.1) of any two locations). One of the neighbor solutions replaces with the current position in the V formation if it improves relative to the current position. Then the best unused neighbor solution is shared by the next solution in the V formation as one of its neighbors' solutions. These processes continue until leader bird (solution) becomes the last bird in the V formation. See pseudocode below.

n = the number of initial solutions (birds)
k = the number of neighbor solutions to be considered
x = the number of neighbor solutions to be shared with the next solution
m = number of tours
K = iteration limit

Pseudocode of MBO:

```
1.  Generate n initial solutions in a random manner and place them on an hypothetical V formation arbitrarily
2.  i = 0
3.  while(i < K)
4.    for (j = 0; j < m; j++)
5.      Try to improve the leading solution by generating and evaluating k neighbors of it
6.      i = i + k
7.      for each solution s_r in the flock (except leader)
8.        Try to improve s_r by evaluating (k–x) neighbors of it and x unused best neighbors from the solution in the
    front
9.        i = i + (k – x)
10.     endfor
11.   endfor
12.   Move the leader solution to the end and forward one of the solutions following it to the leader position
13. endwhile
14. return the best solution in the flock
```

Figure 3.1 Pseudocode of Migrating Birds Optimization [1]

MBO algorithm provides better performance than other metaheuristics such as simulated annealing, tabu search but the neighbor solution possibilities of the birds in the flock is restricted and that is blocker to get better performance.

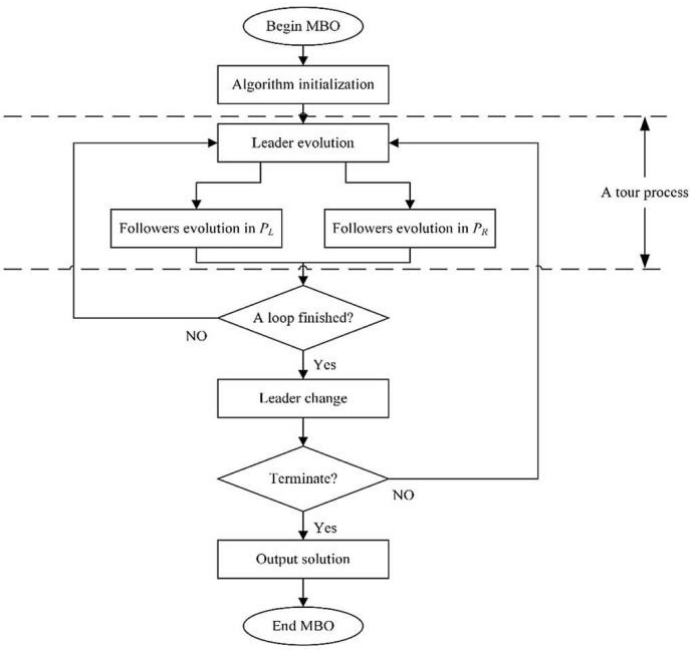

Figure 3.2 Flow chart of the MBO

To get better performance on QAP, we decided to improve the "A tour process" step in the above figure 3.2. We would increase the range of solutions if we could create new solutions by using the solutions we have just like Genetic Algorithms. The GA is a subcategory of the Evolutionary Algorithms. It is another metaheuristic approach inspired by human heredity. The human heredity expresses the inheritance progress of human genetics from parents to their children. Each person gets genetical material from his mother and father. Gens are important to have characteristic features and some illness. The GAs pursues the same logic to solve complex problem. Initially some solutions are selected randomly. That solutions represent of an assignment instance for our project it is a solution of QAP. These steps are occurred by order:

Initial Population: Create a population randomly.

Fitness Function: Each instance gets a cost value which means that it is good solution or not. (Higher value is worse fitness)

Selection: The instances which has better fitness score are selected by priority and pass the next generation.

Crossover: New individuals are created by combining the genetic information of selected individuals. This means mixing genetic material and ensuring diversity.

Mutation: Random changes are made to the genetic information of newly created individuals. This is necessary to enable greater diversity and exploration.

Select a New Population: Individuals (solutions) created because of crossover and mutation form the next generation population.

Stopping Criterion: The steps are repeated until a certain stopping criterion is met. This criterion can be the maximum number of iterations, a certain fitness value, or time constraint.

The goal in the genetic algorithms is to find optimal solution among the individuals in the population by using an evolutionary process. The individuals (solutions) which has great fitness stands for the natural selection and pass the next generations. This improves the quality of final solutions.

Many improvements have been handled on GA to specialize and optimize on complex problems. One of these called Memetic Algorithm can be thought of as an extension or enhancement of the genetic algorithm. The memetic algorithm adds a learning component to the optimization process of the genetic algorithm. This learning component tries to get better solutions by using local search or meta-knowledge in the problem area. That is, the memetic algorithm provides faster and more efficient optimization by adding local search or problem- specific operators to the evolutionary process of the genetic algorithm.

While the memetic algorithm uses the population-based optimization approach of the genetic algorithm, it aims to achieve better results by using local search and learning operators. Therefore, the memetic algorithm improves the performance of the genetic algorithm while also integrating local search strategies. In our project, we will be going to

implement an extension of MA named Multimeme Memetic Algorithm to the MBO which is our actual algorithm.

MMBO
Generate n initial solutions in a random manner and place them on a hypothetical V formation arbitrarily
i = 0
Apply a random hill climber to each individual (solution)
while(i < K)
    for (j = 0; j < m; j++)
        Try to improve the leading solution by generating and evaluating k neighbors of it (**in our imp. k is a meme and takes one of [3,5,7] values**)
        i = i + k
        for each solution s_r in the flock (except leader)
            Try to improve s_r by generating and evaluating (k–x) neighbors of it
            If it cannot improve itself use x unused best neighbors from the solution in the front (**in our imp. x is constant and set to 1**)
            i = i + (k x)
        endfor
    endfor
    move the leader solution to the end and forward one of the solutions following it to the leader position
endwhile
return the best solution in the flock

The hybridization of MBO with MMA will be done in the neighbor creation step of the algorithm which is given as follows:
==========
Neighbor generation (repeated k times for leader, k-x times for others):
    Choose a mate from a tournament selection of size 2,
    Memeplex_to_use ß memeplex of better parent
    Child ← ApplyCrossover(MemeplexToUse_crossover, OriginalParent, Mate)
    Child ← ApplyMutation(MemeplexToUse_mutation, MemeplexToUse_intensity, Child)
    Child ← ApplyLocalSearch(MemeplexToUse_hillclimber, MemeplexToUse_depth, Child)
    Copy the MemeplexToUse to Child
    Mutate meme of Child with a probability of IR

Figure 3.3 Pseudocode of Multimeme Memetic Migrating Birds Optimization

3.1.3 Comparison Metrics

We have different comparison metrics for our project. One of the important metrics is overall cost. Since we want to minimize the overall cost, we compared the cost with previous algorithm which is MBO. When we get minimized cost, maybe it's not a significant improvement. So, one other comparison metric is benchmarking against established methods by applying Wilcoxon Signed-Rank Test. This test indicates if our result worth and significant improvement for established methods and papers.

3.1.4 Data Set or Benchmarks

Our project is applied to a set of benchmark problems obtained from Quadratic Assignment Problem Library (QAPLIB) where it was able to find the best-known solution in most cases. QAPLIB is a widely used resource in the field of combinatorial optimization. It's essentially a collection of benchmark instances specifically designed for testing and comparing the performance of algorithms for

solving the QAP. QAPLIB, established by Burkard, Karisch, and Rendl [10], is a widely recognized benchmark library for the QAP. It offers a standardized set of problem instances with varying sizes and characteristics, enabling researchers and developers to compare and evaluate the performance of different QAP algorithms in a controlled environment.

The library consists of data files containing flow matrix which is an n x n matrix representing the flow or interaction between pairs of facilities and distance matrix which is an n x n matrix representing the distances or costs between pairs of locations.

QAPLIB offers instances with sizes ranging from small (n = 15) to large (n = 315). The instances exhibit diverse characteristics, including symmetric and asymmetric flow and distance matrices are available. They also have different kind of hardness' with varying levels of difficulty are provided, catering to different algorithm testing requirements.

QAPLIB provides a common ground for researchers to compare and evaluate the performance of different algorithms in terms of solution quality, computational time, and robustness. For the development, it provides diverse set of instances allows for testing and refining new algorithms under various conditions, leading to the development of more effective solutions. For tuning, QAPLIB instances can be used to fine-tune algorithm parameters and settings for optimal performance on specific problem types.

Instance files have different names with a naming convention. For example, "lipa40b" and "lipa70a" following the same naming convention:

- **lipa**: This could stand for another set of researchers who created these specific instances, possibly "Li" and "Pa."
- **40** and **70**: These likely refer to the size of the problem, meaning 40 and 70 facilities or locations to be assigned in each instance.
- **b** and **a**: These are likely distinguishing identifiers for different instances within the same size group, potentially indicating variations in problem characteristics or difficulty.

3.2 System Architecture

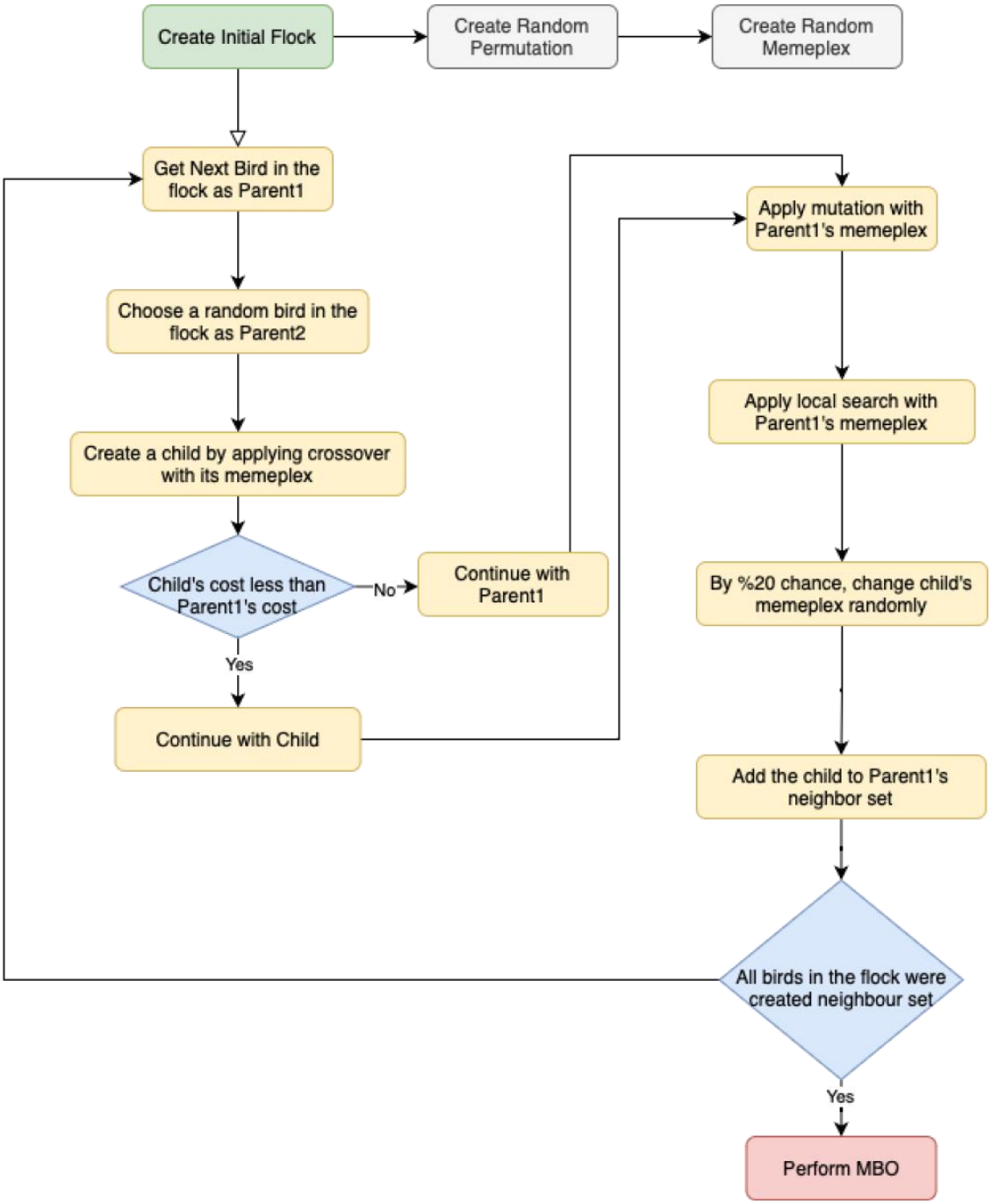

Figure 3.4 Flowchart of Neighbor Creation for MMMBO

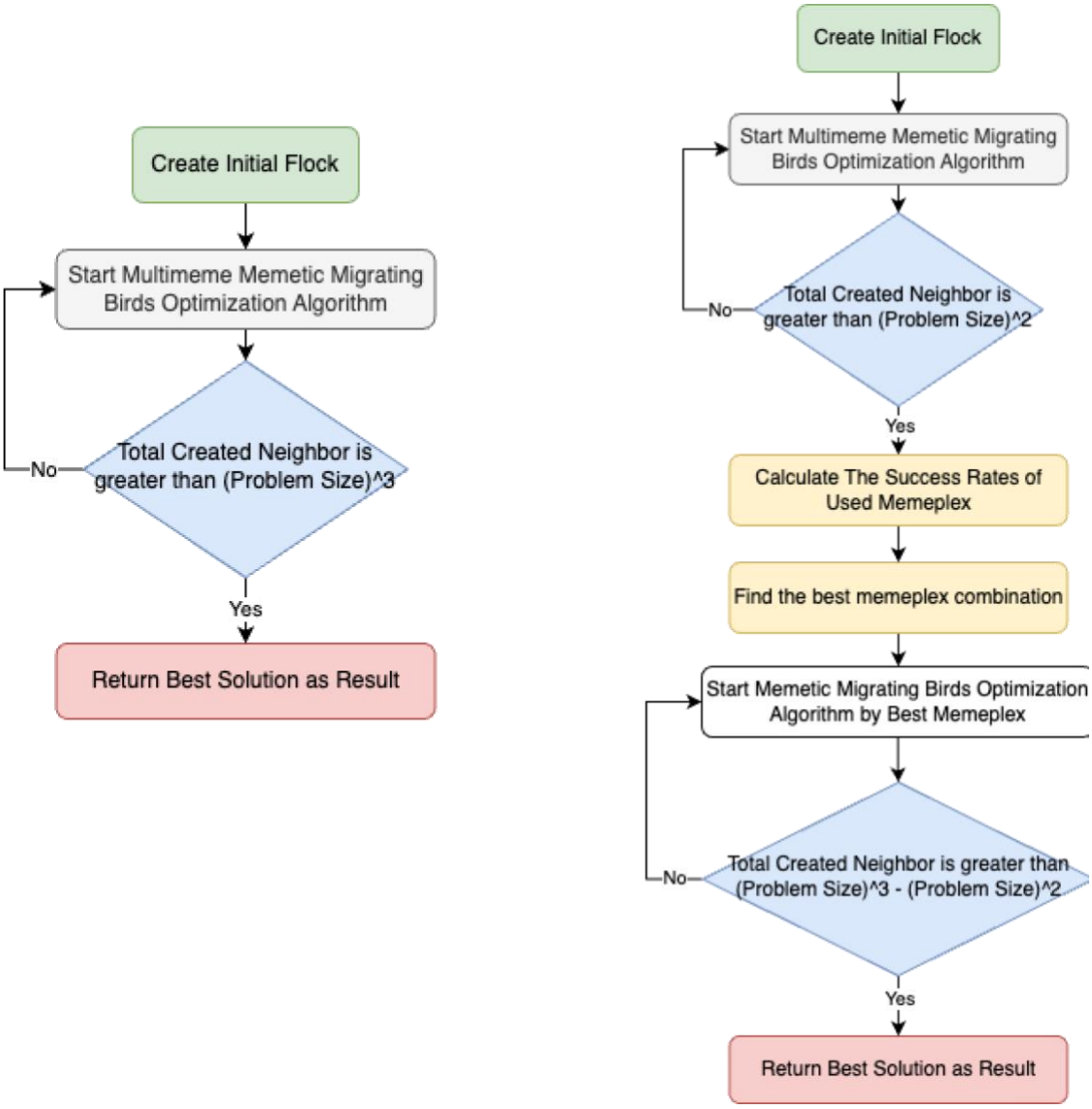

Figure 3.5 Flowcharts of Non-adaptive MMMBO and Adaptive MMMBO

4. TECHNICAL APPROACH AND IMPLEMENTATION DETAILS

4.1 Definition of Solution

All solutions in the solution space represent an ordered permutation. For QAP, it represents which facilities will be built in the cities respectively. But the relationship between the orders may differ. This makes harder to write an algorithm which can find optimal solution in the QAP. Every ordered permutation (solution) has a cost.

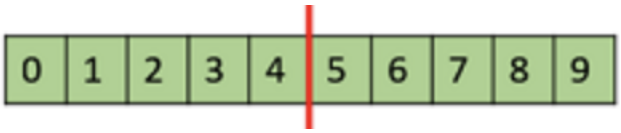

Figure 4.1 Solution Permutation example

Every order change in the permutation may affect the cost by unexpected way. Therefore, solution's permutation cannot be determined how many values are on the right index currently.

4.2 Methodological Approach of Migrating Birds Optimization

General local search strategies use the iteratively improvement the current solutions. Some initial order permutations are created randomly. For every initial solution, neighbor of the solutions is explored and compared with the current solutions. If the neighbor solutions are better, it means that we may get closer to optimal solution. The worse solutions are replaced by the better solutions to achieve an iteratively improvement.

MBO is an optimized local search strategy which uses the V formation of the birds and order change of birds in the formation. Each birds represent a solution permutation for the problem. Fixed size flock are defined to explore that much part of the solution space. For instance, a flock which have 50 birds means that in the solution space, 50 areas on the solution space are exploring the better solutions at the same time.

There is a common issue in local search strategies. Searching on the specific area may lead to stuck in the local optima, because of that the solutions may never

find the global optima. There are some optimizations to avoid this problem. MBO has a unique technique to prevent to stuck in the local optima by its V formation. In the formation, frontier bird shares some of its best neighbors with the next birds. The next bird checks its own neighbors and that shared neighbors to be replaced by the better one.

4.3 Bottleneck of Migrating Birds Optimization

As we mentioned in chapter 4.1, each solution (birds) has an order permutation which represents assignments of the facilities among the cities. Neighbor of the solution is another solution which has some different placement on the permutation array. It is assumed as neighbor because it has common placements. In the visualization of solution space (all possible permutations), two neighbor solutions may have completely different cost value.

Figure 4.2 Heterogenous and homogenous distributed solution space

In the MBO, random mutation operator is used to provide neighbor exploration. It performs basically swapping of the two random placements. Theoretically, such a minor change in the permutation ensures to exploit the solutions in that area. It works with higher performance when the best solutions are collected on the specific area [Figure 4.3.2] of the solution space. Because it will focus on the closer ones. However, it would be slower to explore the area which has optimal solution when the solution space is homogeneous distributed.

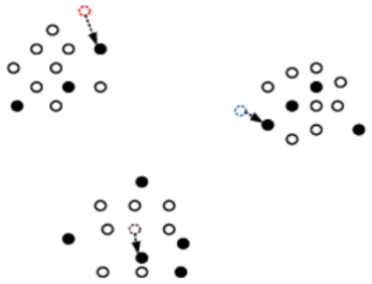

Figure 4.3 Replacement with better solution in local area

Our hybrid approach aims to get reliable and consistent solutions independently of the distribution of the solution space of the problem. Unlike MBO, MMMBO uses advanced neighbor searching process. It does not only swap two placements, but it may change a sequence piece of the permutation by randomly in itself. Our algorithm has a one more improvement for this issue. It does not apply a predefined process for neighbor searching the algorithm uses a co-evolutionary combined process among defined operators [4.4]. It evolves the process by searching and analyzing the obtained neighbor's quality. Thus, algorithm works with consistent performance independently of whether the solution space distribution is heterogeneous or homogeneous.

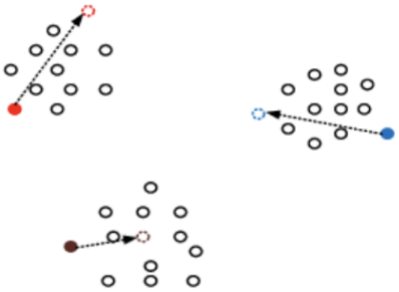

Figure 4.4 Replacement with better solution in far area

4.4 Operators

In the previous chapter, neighbor searching is described briefly and visualized to understand what it aims to do. We mentioned about some operators which search neighbors, so in this chapter, we will discuss about what these operators is and what they do. The definition of operator is to create a permutation by using defined order changing implementation which takes a permutation already exists in population

(flock in MBO) as parameter.

Operator concept is building block of Genetic Algorithm [11]. GA is a metaheuristic inspiring natural selection and inheritance. The main idea is to have a random initialized permutation as population. Then using the population select two permutations as parents, apply crossover operator and acquire a different permutation. Then apply mutation to that child permutation to increase possible explorations. Add the obtained child permutation to your population. As you can see below the illustration of GA, operators are meant to diversify the solutions existing.

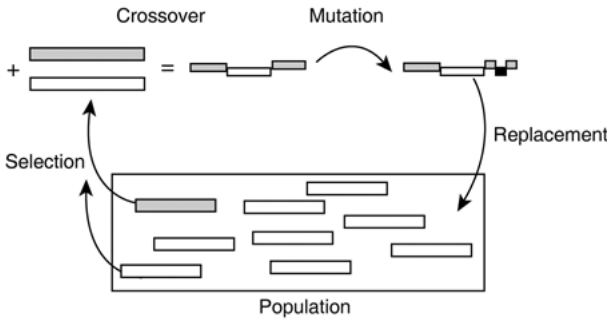

Figure 4.5 Representation of genetic algorithm

There are 8 different operators type in our hybrid model: 3 crossover, 3 mutation and 2 local search operators. Besides that, there are 10 different parameter type for mutation operators and local search operators: 5 mutation intensity, 5 local search depth parameters.

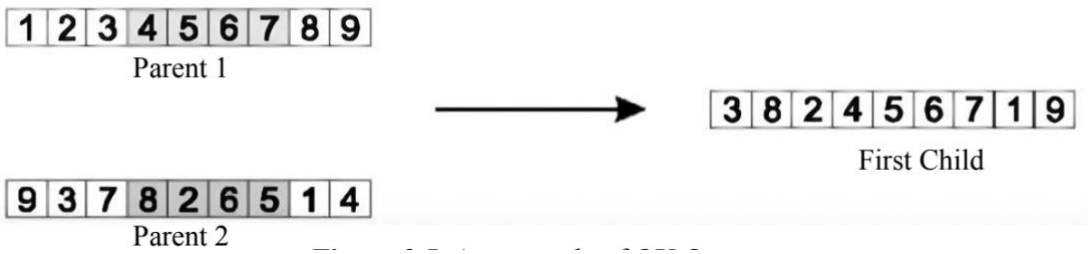

Figure 4.6 Representation of Order Crossover

Order Crossover Operator:

- Choose a range in Parent1 permutation randomly.
- Put the piece of permutation to the Child's empty permutation.
- For the remaining part, adding placements from Parent2 to the child one by one.

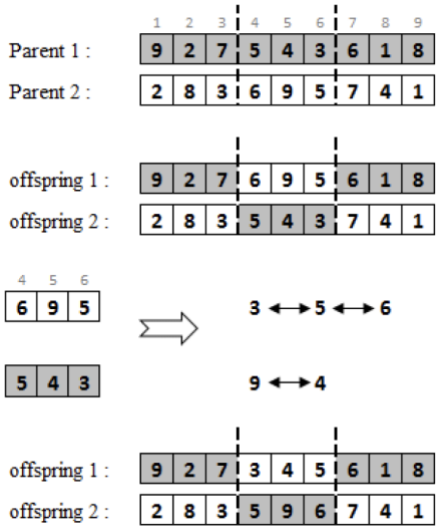

Figure 4.7 Representation of PMX

Partially Matched Crossover Operator:

- Copied the Parent1's permutation to the Child.
- Choose a range in Parent2 permutation randomly.
- Put the piece of the selected permutation to the Child's permutation.
- To avoid duplicated index in the created child's permutation, use matching table for the chosen range of parent1 and parent2.

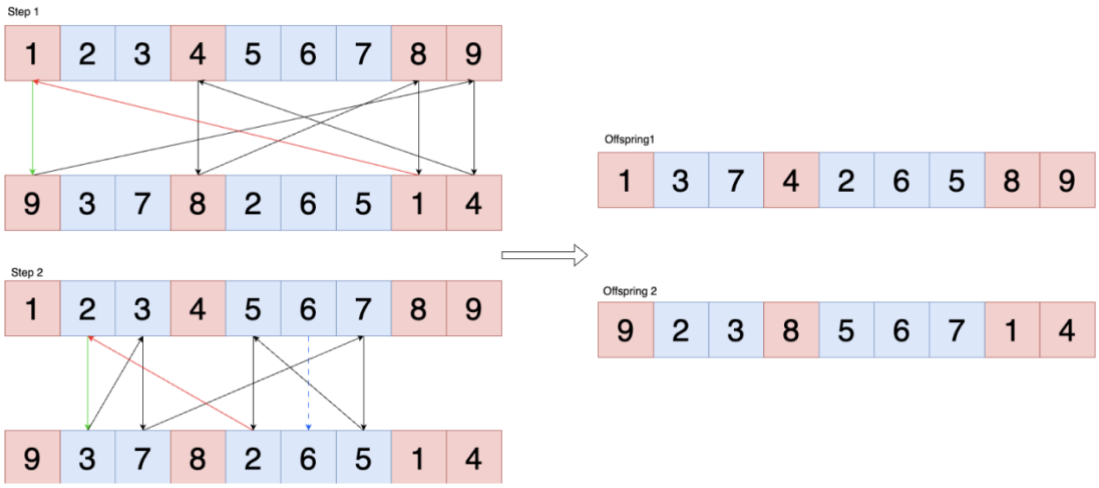

Figure 4.8 Representation of CX

Cycle Crossover Operator:

- Find a cycle between Parent1 and Parent2
- Put the index in the Parent1 which exist in the cycle to the Child.
- Put the index in the Parent2 which does not exist in the cycle to the remaining indexes of the Child.

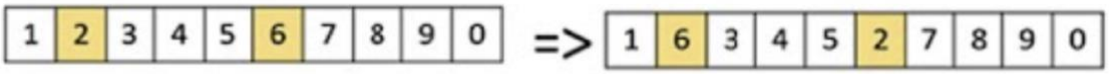

Figure 4.9 Representation of SwapRandom

SwapRandom Mutation Operator:

- Choose two random indexes in the permutation.
- Swap them.
- Repeat the process up to the 5* α (1,2,3,4,5) times.

SwapBest Mutation Operator:

- Choose two random indexes in the permutation.
- Swap them.
- Stop the mutation if current cost of the permutation better than the child.
- Repeat the process up to the 1000*(α^3) (8,64,216,512,1000) times.

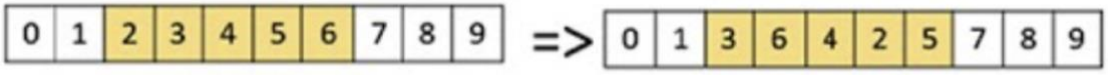

Figure 4.10 Representation of SwapBest

ScrambleSwap Mutation Operator:

- Choose a range in the permutation.
- Swap the index in the range randomly 5* α (1,2,3,4,5) times.
- Repeat the process up to the 5* α (1,2,3,4,5) times.

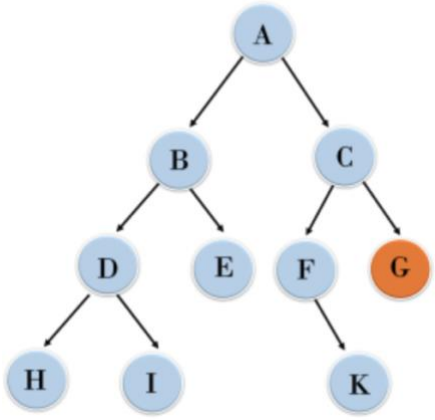

Figure 4.11 Representation of SwapFirstII

SwapFirstII Local Search Operator:

- Swap two indexes of the permutation randomly
- Stop the local search if current cost of the permutation is better than the Child.
- Repeat the process up to the 5* α.
- Repeat the process up to the depth of local search (1,2,3,4,5)

SwapBestII Local Search Operator:

- Swap two indexes of the permutation randomly
- Repeat the process up to the 5* α.
- Repeat the process up to the depth of local search (1,2,3,4,5)
- Change the permutation of child by the explored local neighbors of the permutation.

4.5 Methodological Approach of Multimeme Memetic Algorithm

Local search strategies aim to improve the quality of the solution pool as we talked about earlier chapters. In the genetic algorithm, the performance of the algorithm depends on the kind of the problem dataset. Because it always uses the same crossover and mutation operator combination set to search all solution space. To satisfy the issue, released an optimization of the GA called Multimeme Memetic Algorithm [3] which does not use a specific operator combination. Unlike that, it evolves the quality of operator combination in the solutions. Each solution stores its operator combination array. The neighbor creation of the solution depends on that operator combination array. The flow of the neighbor search is the same but in the implementation of the operators different and they may give extremely different impact at creating more effective neighbors. To implement this co-evolutionary searching, each solution keeps a 5 sized array called memeplex. The first index of the array is for crossover operator. The second index of the array is for mutation operator. The third one is for local search operator and the fourth and fifth indexes are for mutation intensity and depth of local search parameters.

Better memeplex array will live longer in the population because each iteration we are replacing the birds (solutions) with the better neighbors. Each neighbor inherits the memeplex data of its parent. Therefore, the population will have the most successful memeplexes as time goes by. This will help the algorithm to increase its succession to searching the neighbors.

4.6 Neighbor Creation Process of MMMBO Algorithm

Now, we have mentioned the ideas and the assets which we used to design our algorithm. In this chapter, you will see the actual implementation of searching for the neighbor solutions and improvement of the solution in the population. MBO uses only mutation operator with fixed mutation intensity parameter to create new neighbors. In our hybrid optimization we implemented advanced neighbor creation on migrating birds' optimization instead of basic mutation operator.

Each bird in the flock selects a mate from it, a child is created by applying a crossover operation in the first index of memeplex. Child is applied a mutation operator by the second index in the memeplex array. Then the child's neighbors are searched to check if there are better solutions near the child.

4.7 Adaptive Multimeme Memetic Migrating Birds Optimization

Our MMMBO algorithm has had a satisfactory performance for some instances, but the results were not very consistent as we expected. We observed that our co-evolutionary memetic searching mechanism was explored on the huge area range. Because each operator combination implements a separate way. In some point of the iteration, the current results may need successfully exploit the searching area. However, the successful memeplex may not exploit the area as we expected to do, they may continue to explore the areas of the different spot of the solution space. This issue can cause the algorithm to spend its performance and effort for redundant action. We researched and discussed to find a solution about this exploration and exploitation dilemma.

To solve this bottleneck, we made adaptive our algorithm flow. The main idea of this implementation is to give a limited time to the algorithm to introduce and evolve the successful memeplexes for the given problem. Meanwhile, succession rate of every operator which is performed are analyzed. After the flock is evolved and takes shape of the problem, the best memeplex (best operator combination array) is calculated. And for the remaining time of the algorithm, the neighbor searching process works according to that best memeplex data. This adaptiveness provides us to get much consistent and reliable results at the end of the algorithm.

5. EXPERIMENTAL STUDY

5.1 Overview of Experimental Setup

5.1.1 Software Environment

- The experiments were conducted in a Java-based environment, specifically utilizing Java 21 for algorithm implementation and execution.
- Git version control was employed for source code management, allowing for version tracking and collaborative development.
- GitHub served as the platform for hosting the project repository, facilitating version control collaboration and providing accessibility to the codebase.

5.1.2 Libraries and Dependencies

- The algorithm implementation leveraged the latest stable version of Apache POI for Java. Apache POI was used to write and manage experimental results in Excel files.

5.1.3 Execution Platform

- The experiments were executed on a standard computing environment without any specific hardware requirements.
- Java Virtual Machine (JVM) compatibility and availability were ensured for running the Java-based algorithms.

5.2 Comparison of Algorithms

Figure 5.1 Comparison of MBO, MMBOv1, MMBOv2 in wil50 instance

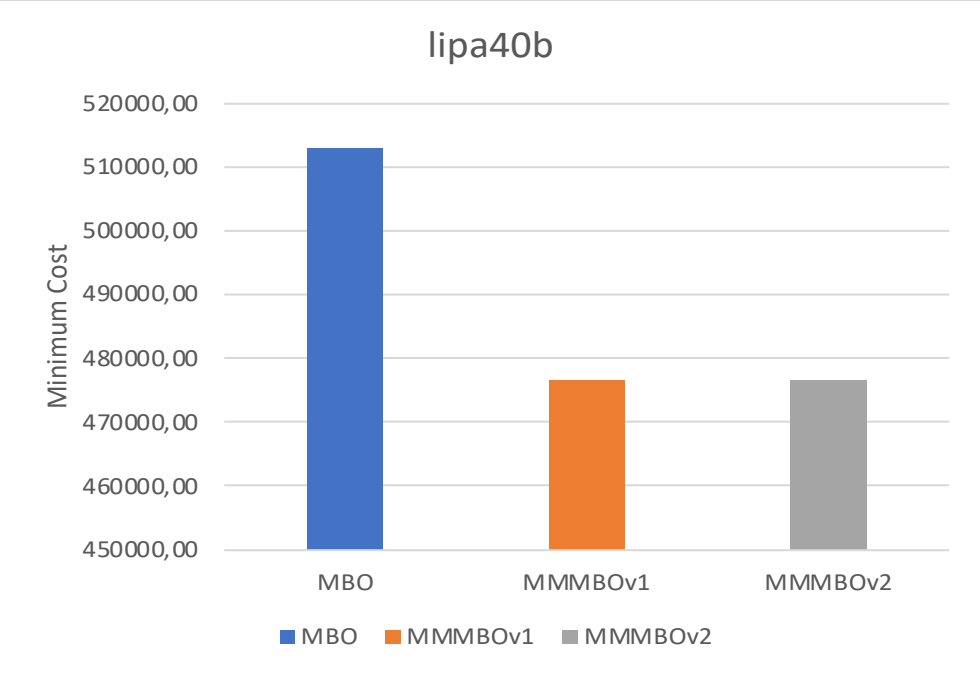

Figure 5.2 Comparison of MBO, MMBOv1, MMBOv2 in lipa40 instance

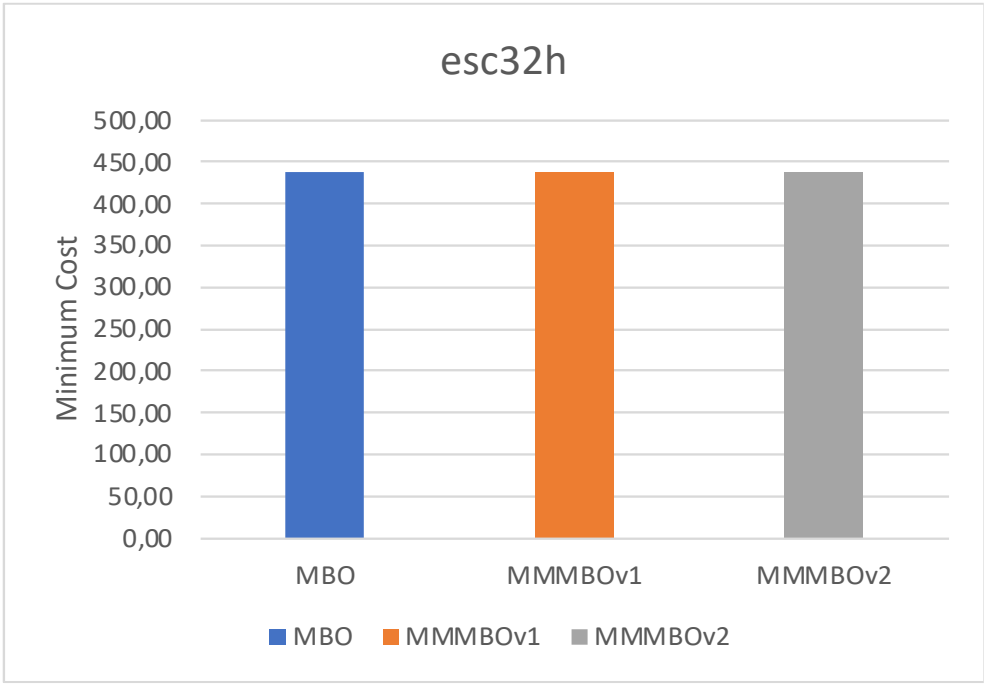

Figure 5.3 Comparison of MBO, MMBOv1, MMBOv2 in essc32h instance

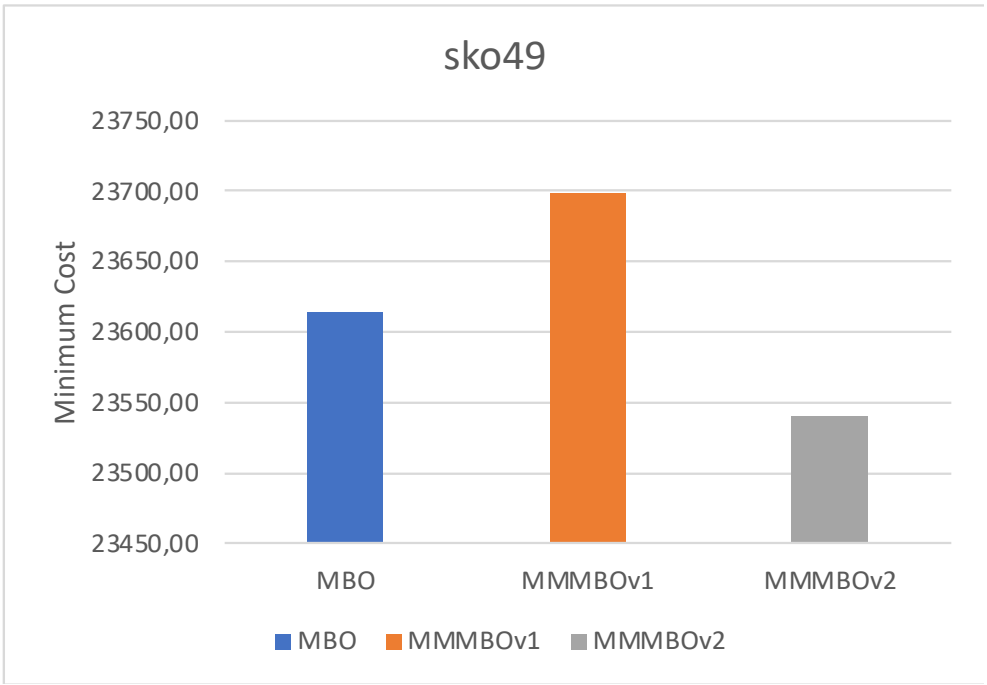

Figure 5.4 Comparison of MBO, MMBOv1, MMBOv2 in sko49 instance

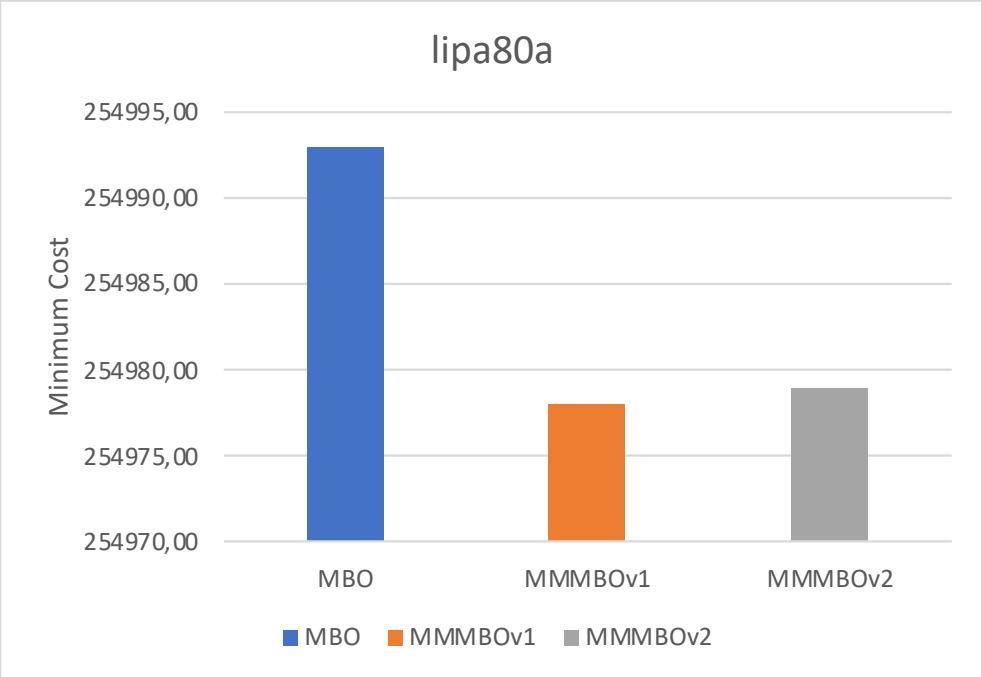

Figure 5.5 Comparison of MBO, MMBOv1, MMBOv2 in lipa80a instance

5.3 Wilcoxon Test

The Wilcoxon test, also known as the Wilcoxon signed-rank test, is a non-parametric statistical method employed to compare two related samples. In the context of algorithmic performance evaluation, the test is utilized to determine whether there are significant differences between the performance of two algorithms across a set of instances.

The test involves ranking the absolute differences between paired observations, assessing whether the distribution of positive and negative ranks is symmetrical. Through this process, the Wilcoxon test provides insights into whether one algorithm consistently outperforms another or if there is a statistically significant difference in their performance.

In our context, the notation $>$ ($<$) and $\geq$ ($\leq$) is employed to interpret the test results, signifying statistically significant and non-statistically significant performance differences, respectively. This rigorous statistical analysis aids in drawing robust conclusions about the relative effectiveness of algorithms under consideration, informing decisions for algorithm refinement and optimization.

Instance ID	MBO		MMMBOv1
	MMMBOv1	MMMBOv2	MMMBOv2
esc32e	-	-	-
esc32f	-	-	-
esc32g	-	-	-
esc32h	>	>	>=
lipa40b	>=	<	>
sko49	>	<	<=
wil50	<=	<	>
tai60b	<=	<	>
tai64c	<=	<=	<=
esc64a	-	-	-
lipa70a	<	>	<=
lipa80a	<	<	>=

Table 5.1 Wilcoxon test results

- Wilcoxon signed-rank test was conducted to compare MMMBOV1, MMMBOv2 and MBO performance across multiple instances.
- All tree algorithms found the optimal solution always for instances: esc32e, esc32f, esc32g and esc64a.
- Instances sko49, wil50, and tai60b show significant performance differences, favoring MMMBOv1.
- Tai64c, lipa70a, and lipa80a exhibit no statistically significant variance between MMMBOv1 and MMMBOv2.
- Performance discrepancies highlight the need for understanding specific instance characteristics for algorithm refinement.
- Optimization efforts should consider the insights gained from the Wilcoxon test results to enhance MMMBO algorithm outcomes.

In conclusion, we can say we've improved MBO for solving the QAP.

6. BENEFITS AND IMPACT

6.1 Scientific Impact

This project aims to improve solution for QAP by combining different algorithms and their various operators. So that it can illuminate other researchers' vision. We believe that it will bring out new ideas to most of the researchers with this study. It has a great potential to be a scientific paper.

6.2 Potential Impact on New Projects

The potential impact of improving the Quadratic Assignment Problem (QAP) solution through the hybridization of Migration Birds Optimization and memetic algorithms could be significant for new projects. By leveraging these hybrid algorithms, there is the prospect of achieving enhanced optimization capabilities, which can positively influence the efficiency and effectiveness of various projects. This hybrid approach may lead to quicker problem-solving, better resource utilization, and improved overall project outcomes. Additionally, the innovation in problem-solving methodologies can contribute to a more robust and adaptable framework, providing a valuable tool for addressing complex challenges in diverse project scenarios. Ultimately, the integration of advanced optimization techniques has the potential to elevate the success and impact of new projects across different domains.

6.3 Economic/Commercial/Social Impact

The project's result will yield an academic study beneficial for researchers in the QAP domain, enhancing understanding of memetic algorithms and MBO domains. Additionally, the improved solution to the QAP can have a noteworthy economic impact by optimizing processes, fostering efficiency, and potentially leading to resource savings and economic benefits in various applications.

6.4 Impact on National Security

This project is based on academic research, therefore the impact of national security does not apply to this project.

7. CONCLUSION AND FUTURE WORK

Our work to enhance the Migration Birds Optimization (MBO) algorithm within the framework of solving the Quadratic Assignment Problem (QAP) have resulted in significant strides, particularly when considering the integration of operators in the hybridization process. The refinement of MBO not only represents a meaningful advancement in addressing the challenges of large-scale combinatorial optimization problems but also demonstrates the effectiveness of combining multimeme memetic algorithms with MBO.

Key to our success is the thoughtful integration of various operators in the hybridization process, such as local search, mutation, and crossover operators like Partially Mapped Crossover (PMX), Order Crossover (OX), and Cycle Crossover (CX). These operators play a precious role in shaping the algorithm's behavior, reflecting exploration and exploitation of the solution space. The inclusion of local search operators contributes to fine-tuning solutions, refining the quality of assignments produced by the algorithm.

By integrating these diverse operators, we aimed not only to enhance the overall efficiency of MBO but also to utilize the strengths of multimeme algorithms in promoting diversity and robustness in the solution space exploration. This adapted metaheuristic solution, characterized by the hybridizing of MBO and multimeme memetic algorithms, stands as a testament to our commitment to advancing not only the smoothness of algorithmic techniques but also their practical utility in solving complex optimization problems.

As we reflect on the project's outcomes, the improved MBO algorithm, enriched by the hybridization of operators, opens new ideas for future research in combinatorial optimization. The interaction of local search, mutation, and crossover operators showcases the adaptability of our approach, providing valuable insights for researchers and practitioners alike. The potential applications of these refined algorithms extend beyond the QAP domain, offering promising road for tackling optimization challenges across various industries. Our work not only contributes to the ongoing evolution of heuristic algorithms but also lays the groundwork for continued advancements in solving complex optimization problems with solid implications for real-world problem-solving.

REFERENCES

[1] Duman, Ekrem & Uysal, Mitat & Alkaya, Ali. Migrating Birds Optimization: A New Metaheuristic Approach and Its Application to the Quadratic Assignment Problem, Information Sciences, June 2012.

[2] Harris, Matthew & Berretta, Regina & Inostroza-Ponta, Mario & Moscato, Pablo. (2015). A memetic algorithm for the quadratic assignment problem with parallel local search, May 2015

[3] Meneses, H., and Inostroza-Ponta, M.: 'Evaluating memory schemas in a Memetic Algorithm for the Quadratic Assignment Problem'. Proc. Computer Science Society (SCCC), 2011 30th International Conference of the Chilean2011 pp.

[4] Dreo, J. Petrowski, A. Siarry, P., Taillard, E., "Metaheuristics for Hard Optimization", 1st Edition, Springer, New York. September 2005.

[5] Ochoa, Gabriela & Hyde, Matthew & Curtois, Timothy & Vázquez Rodríguez, José Antonio & Walker, James & Gendreau, Michel & Kendall, Graham & Mccollum, Barry & Parkes, Andrew & Petrovic, Sanja & Burke, Edmund. (2012). HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search, April 2012.

[6] Berretta, R., & Moscato, P.: 'The number partitioning problem: an open challenge for evolutionary computation?': 'New ideas in optimization' (McGraw-Hill Ltd., UK., 1999), pp. 261-278

[7] Berretta, R., & Rodrigues, L. F.: 'A memetic algorithm for a multistage capacitated lot- sizing problem', International Journal of Production Economics, 2004, 87, (1), pp. 67-81

[8] Buriol, L., França, P. M., & Moscato, P.: 'A new memetic algorithm for the asymmetric traveling salesman problem', Journal of Heuristics, 2004, 10, (5), pp. 483-506

[9] Moscato, P., Mendes, A., and Berretta, R.: 'Benchmarking a memetic algorithm for ordering microarray data', Biosystems, 2007, 88, (1), pp. 56-75

[10] Burkard, R.E., Karisch, S., & Rendl, F.: 'QAPLIB-A quadratic assignment

problem library', European Journal of Operational Research, 1991, 55, (11), pp. 115-119

[11] Kirkpatrick, S. G., C. D. Delatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing", May 1983.

[12] Battiti, R., Tecchiolli,G. ,"Simulated Annealing and Tabu Search in the Long run: A comparison on QAP Tasks" ,Computers Mathematical Applications 28,1-8, September 1994.

[13] Drugan, Madalina & Talbi, El-Ghazali. Adaptive Multi-operator MetaHeuristics for Quadratic Assignment Problems. Advances in Intelligent Systems and Computing, July 2014