INTELLIGENT SOFTWARE TOOL FOR SCHEDULING NURSES IN HOSPITALS: A COMPARATIVE STUDY OF TWO ANT COLONY OPTIMISATION HYPER-HEURISTIC SCHEMES

BY

EHIREMHEN OFURE
(20CG028069)

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES, COLLEGE OF SCIENCE AND TECHNOLOGY, COVENANT UNIVERSITY OTA, OGUN STATE.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE BACHELOR OF SCIENCE (HONOURS) DEGREE IN COMPUTER SCIENCE.

JUNE 2024

CERTIFICATION

I hereby certify that this project was carried out by Ehiremhen Ofure (20CG028069) in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ogun State, Nigeria, under my supervision.

Dr. Adubi Stephen	______________________________
Supervisor	**Signature and Date**
Professor Olufunke O. Oladipupo	______________________________
Head of Department	**Signature and Date**

DEDICATION

I dedicate this project to God, without whom none of this would have been possible. I also dedicate this work to my supervisor who has been there to help me whenever I needed it, and to my friends and family as well, for the support.

ACKNOWLEDGEMENTS

My most profound gratitude goes to Almighty God, who has kept me and sustained me right from the very start of my degree up until this point. To Him be all the glory.

I am extremely grateful to my parents, Mr and Dr Ehiremhen for their sacrifices, contributions, and prayers towards the completion of this degree. I also want to appreciate my amazing siblings, for their pieces of advice, support and encouragements during the course of this project and degree. I am extremely privileged to have them in my life.

My sincere gratitude also goes to my hardworking supervisor, Dr Adubi Stephen, who is always there when I need him. Thank you so much for your excellent supervision and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1. BACKGROUND INFORMATION

The healthcare industry, particularly the management of nursing staff in hospitals, has always been a critical and challenging domain. With the increasing demand for health-care services, efficient scheduling of nurses has become a pressing issue. Properly scheduling the nursing staff has a great impact on the quality of health care (Oldenkamp, 1996), the recruitment of nursing personnel, the development of a nursing budgets and various other functions of the nursing service (Cheang et al., 2003). Historically, nurse scheduling has been a manual and time-consuming task, often leading to suboptimal schedules and resource allocation. The advent of intelligent software tools presents a promising solution to this long-standing problem.

Despite the critical role of nurse scheduling, various challenges persist within this domain. These challenges include the need to balance to satisfy various hard constraints such as legal requirements, staff preferences, and institutional regulations, and as many soft constraints as possible, such as minimal nurse demands, “day-off” requests, personal preferences, etc. Traditional scheduling methods often struggle to efficiently navigate these multifaceted constraints, resulting in suboptimal nurse schedules that can lead to decreased staff satisfaction, increased burnout, and ultimately, compromised patient care. Additionally, the ever-changing nature of healthcare environments exacerbates the complexity of nurse rostering, demanding adaptive and intelligent solutions capable of optimizing schedules in real-time.

Despite the growing interest in intelligent scheduling tools, there’s a notable research gap concerning the exploration and comparison of specific optimization techniques, such as Ant Colony Optimization (ACO) hyper-heuristic schemes, for nurse scheduling. While ACO has been successfully employed in various optimization domains, its application in nurse scheduling remains relatively underexplored. This project aims to address this gap by proposing an intelligent software tool that utilizes two distinct ACO hyper-heuristic schemes tailored for nurse scheduling. Drawing inspiration from successful applications in related fields, this proposed solution seeks to demonstrate its effectiveness in navigating the intricate constraints of nurse rostering and overcoming the specific challenges encountered in nurse scheduling.

Over the years, various scheduling algorithms have been employed by researchers to solve this problem. These methods include Tabu Search (Burke, Kendall, & Soubeiga 2003), Variable Neighborhood Search (Bilgin et al., 2012), Simulated Annealing (Brusco & Jacobs, 1995), Ant Colony Optimization (Gutjahr, & Rauner, 2007), Branch-and-price Algorithm (Maenhout, & Vanhoucke, 2010), Harmony Search (Awadallah et al., 2013) and Scatter Search (Curtois & Berghe, 2010). A comprehensive analysis reveals that the existing works often fall short in effectively addressing the intricacies of nurse rostering problems, and whose main disadvantage lies in its high computational complexity which limits their application only to small size instances, and when attempted to be scaled, either cost too much computation, or have difficulty in constraint handling. Therefore, alternative optimization methods, namely heuristics and metaheuristics have been developed in order to find suboptimal solutions of good quality in a reasonable time. Existing works in this field have primarily focused on traditional scheduling methods or have explored single heuristic or metaheuristic approaches. The opportunity lies in the comparative evaluation of multiple ACO hyper--heuristic schemes, which has not been extensively addressed in the context of nurse scheduling. Methodologically, this study presents an opportunity to fill this gap by rigorously comparing the performance of distinct ACO hyper-heuristic schemes in the context of nurse scheduling.

The hypothesis of this study is that the comparative analysis of two ACO hyper-heuristic schemes will reveal significant differences in their effectiveness for nurse scheduling. This approach is significant as it aims to provide empirical evidence to support the selection of the most suitable ACO hyper-heuristic scheme for nurse scheduling, thereby improving the efficiency and fairness of the scheduling process. In summary, this research seeks to address the existing gaps in nurse scheduling methodologies and to contribute to the advancement of intelligent software tools for nurse scheduling, ultimately enhancing both staff satisfaction and the quality of patient care.

1.2. STATEMENT OF THE PROBLEM

Although the interest in intelligent scheduling tools is on the rise, there exists a research gap in the exploration and comparative analysis of specific optimization techniques, such as Ant Colony Optimisation (ACO) hyper-heuristic schemes, for nurse scheduling. While ACO has been applied to various optimization problems, its application to nurse scheduling remains relatively unexplored. This project aims to bridge this gap by proposing an intelligent software tool that utilizes two distinct ant colony optimization hyper-heuristic schemes for nurse scheduling. Drawing inspiration from successful applications of similar approaches in related domains, this proposed solution seeks to demonstrate its efficacy in handling the intricate constraints of nurse rostering and addressing the specific challenges within the nurse scheduling domain.

To address the identified research gap, this study proposes an innovative solution through a comparative analysis of ACO hyper-heuristic schemes. By leveraging insights gained from related domains and successful applications of similar approaches, the research aims to unveil the nuanced performance differences between the identified ACO hyper-heuristic schemes, providing empirical evidence to support the selection of the most suitable ACO hyper-heuristic scheme for nurse scheduling. The objective is to develop an intelligent software tool specifically tailored for nurse scheduling, leveraging two different variations of the Ant Colony Optimization algorithm, in order to juxtapose the results gotten from both of them and prove which is best suited for the problem, thereby enhancing adaptability and efficiency within hospital settings. Through this exploration, the study seeks to contribute valuable insights and advancements that can significantly improve patient care, staff satisfaction, and the overall operational efficiency of healthcare institutions.

1.3. AIM AND OBJECTIVES OF THE STUDY

The aim of this study is to develop an intelligent software tool to solve the Nurse Rostering Problem using two ant colony optimization hyper-heuristic schemes. The objectives of this study include:

i To extensively review and study existing ACO metaheuristic schemes to get a better understanding of how the general algorithm works towards applying them as hyper--heuristics.

ii To implement two selected ACO hyper-heuristic schemes.

iii To develop an intelligent software tool for nurse scheduling using the selected ACO hyper-heuristic schemes.

iv To test run the ACO hyper-heuristic schemes on the NRP instances.

v To evaluate and compare the results of the two ACO hyper-heuristic schemes and identify the most suitable scheme for nurse scheduling.

vi To compare the most suitable ACO scheme gotten from the previous evaluation with other existing heuristic schemes in literature.

1.4. METHODOLOGY

The tools and techniques I intend on adopting to achieve the above objectives include:

i To achieve objective 1, there is going to be a comprehensive review on existing literature related to ACO meta-heuristic schemes.

ii To achieve objective 2, implementing the selected ACO hyper-heuristic schemes in the Java language.

iii To achieve objective 3, developing the software tool on the Hyper Heuristic Flexible Framework(HyFlex) and using swing GUI for the interface.

iv To achieve objective 4, implementing the developed tool in a simulated environment and obtaining the objective function values of the hyper-heuristic schemes on the NRP instances.

v To achieve objectives 5 and 6, employ statistical methods to compare the results obtained from the two ACO hyper-heuristic schemes, evaluating them using metrics like Formula one scoring, Box-plot visualization and average normalization and identifying the most suitable scheme for nurse scheduling.

Table 1.1: Objectives-Methodology Mapping Table

S/N	OBJECTIVES	METHODOLOGY
1	Reviewing of already existing Ant Colony optimization meta-heuristic schemes, in the context of nurse scheduling.	**Extensive Literature Review** Review of literature on already Ant Colony meta-heuristic schemes in order to get a deeper understanding.
2	Implementation of two selected ACO hyper-heuristic schemes	**Software Development and Algorithm implementation** Adoption of the JAVA programming language to implement the two Ant Colony hyper-heuristic schemes.
3	Development of an intelligent software tool for solving the nurse rostering problem using the selected and implemented ACO hyper-heuristic schemes	**Software Development** developing the software tool on the Hyper Heuristic Flexible Framework(HyFlex), using the JAVA swing UI for the interface.
4	Test-run the ACO hyper-heuristic schemes on the Nurse Rostering Problem instances.	implementing the developed tool in a simulated environment and obtaining the objective function values of the hyper-heuristic schemes on the NRP instances.
5	Evaluation and comparison of the results of the two ACO hyper-heuristic schemes and identify the most suitable scheme for nurse scheduling. .	employ statistical methods to compare the results obtained from the two ACO hyper-heuristic schemes, evaluating them using metrics like Formula one scoring, Box-plot visualization and average normalization and

		identifying the most suitable scheme for nurse scheduling.
6	Comparison of the most suitable ACO scheme gotten from the previous evaluation with other existing heuristic schemes in literature .	Same methodology as above

1.5. SIGNIFICANCE OF THE STUDY

The significance of this study lies in its potential to revolutionize nurse scheduling practices within hospital management, offering substantial benefits to healthcare institutions, nursing staff, and ultimately, the quality of patient care. The study is important for the following reasons:

i The study is important in the sense that it will contribute to the advancement of knowledge, and help to bridge the knowledge gap in the field by comparing the performance of two ACO hyper-heuristic schemes inorder to prove which one provides a more optimal schedule, since existing works in the field have primarily focused on traditional scheduling methods or single heuristic or optimization techniques.

ii Healthcare managers and decision-makers stand to benefit from the study's recommendations and guidelines for implementing the intelligent software tool. The insights gained from the research can inform strategic decision-making on the applicability of advanced scheduling technologies like ACO hyper-heuristic schemes in solving the nurse rostering problem.

1.6. LIMITATION OF THE STUDY

While Ant Colony Optimization (ACO) meta-heuristics offer a powerful approach to solving Nurse Rostering Problems (NRPs), they are not without limitations. These limitations can become particularly pronounced in large-scale NRP scenarios that are heavily constrained. ACO algorithms can struggle to effectively handle these complex constraints, potentially leading to suboptimal solutions that don't fully satisfy all requirements. Additionally, achieving optimal convergence with ACO, where the absolute best solution is found, can be challenging, especially with highly constrained problems. Finally, tuning the internal parameters of ACO can be a time-consuming process. Since our proposed hyper-heuristic schemes are built upon these ACO meta-heuristics, they will inevitably inherit some of these limitations.

1.7. ORGANIZATION OF THE PROJECT

Chapter One of the project contains an introduction to the project: an explanation of the project, the problem I am trying to solve, problems with existing solutions and methods, the method of implementation, the significance of the study, and the limitations. Chapter Two describes the existing systems and methods related to the project topic, the methodology, algorithm, and techniques used in related systems, as well as related works. Chapter Three describes the analysis and system design. Chapter Four shows the implementation of the system in detail and the results obtained. Chapter Five summarises the project and gives recommendations, suggestions, conclusions, and references.

CHAPTER TWO

LITERATURE REVIEW

2.1. PREAMBLE

As the demand for quality healthcare services continues to rise, the need for intelligent software tools to streamline and optimize nurse scheduling processes becomes increasingly evident.

This chapter delves into a comprehensive exploration of existing literature, theories, and methodologies relevant to nurse scheduling, with a specific focus on the application of Ant Colony Optimization (ACO) hyper-heuristic schemes. Through a comparative study of two ACO hyper-heuristic schemes, this research aims to advance scheduling practices in healthcare. By meticulously reviewing scholarly works, this chapter identifies gaps, challenges, and opportunities in nurse scheduling, paving the way for empirical investigation and evaluation of an intelligent software tool.

2.2. REVIEW OF THE NURSE ROSTERING PROBLEM(NRP)

The Nurse Rostering Problem (NRP), also known as the Nurse Scheduling Problem (NSP), was formally introduced in its current form in 1976 through parallel publications by Miller et al. and Warner (Chong et al., 2022). This problem involves assigning shifts and holidays to nurses while considering their individual preferences and constraints, along with the hospital's requirements. Various constraints, both hard and soft, shape the scheduling process, with hard constraints leading to invalid schedules if not met. Soft constraints, on the other hand, influence the quality of solutions without rendering them invalid. Coverage requirements, like, nurse demand per day, per skill, or per shift, are normally considered as hard constraints, while constraints that involve time requirements are usually regarded as soft constraints. Usually, nursing officers spend a substantial amount of time developing rosters especially when there are many staff requests, and where even more time can be consumed in handling ad-hoc changes to current duty rosters. Because of tedious and time-consuming manual scheduling, and for various other reasons, the nurse rostering problem (NRP) or the nurse scheduling problem (NSP) has attracted much research attention (Cheang, Li, Lim, & Rodriguez, 2003).

Nurse Rostering Problems is a complex combinatorial optimisation problem, under the variant of staff scheduling problems and is known to be NP-hard, indicating its computational challenge. This means that there are many thousands of possible schedules, and the staff members may even be unable to determine whether or not there exists a solution at all. As the number of nurses, days in the schedule, and constraints increase, the NRP experiences a combinatorial explosion that makes it challenging to produce an optimal solution (Nasiri and Rahvar, 2017).

Hospitals typically operate with three distinct work shifts—day, evening, and night—each spanning 7-9 hours, depending on local regulations. For instance, some hospitals implement three equal time spans of 8 hours for each shift, while others have shifts of 7, 8 and 9 hours, adjusting workload distribution throughout the day. Various local rules and policies, such as minimum nurse staffing requirements per shift, limits on daily and weekly work hours, and specific skill requirements, serve as constraints. The primary objective is to devise a cost-effective nurse-to-shift assignment that adheres to these constraints, encompassing both hard and soft constraints. Hard constraints are non-negotiable conditions that must be satisfied for a feasible solution, whereas soft constraints, although not mandatory, incur penalties if violated. Given their incorporation as costs within the objective function, we predominantly prioritize addressing hard constraints over soft constraints. In real world nurse rostering settings, the problems are nearly always overconstrained, so, it is unlikely to find a single schedule that perfectly adheres to all of them. An objective function helps us choose the schedule that comes closest to achieving our desired outcomes and is usually based upon some criteria set by the hospital mamagement. Hospitals usually seek to optimize specific types of objective functions when solving the NSP. For instance, objective functions related to staff satisfaction and balanced workload have received major attention in recent years. The most common staff satisfaction objective functions are typically related to the maximization of nurses' preferences, where each preference of a single nurse can be related to desirable working shifts and days off during the planning horizon (Otero et al., 2023).

The mechanisms and performance of the existing solution methodologies applied to the benchmark and real world NRPs can be classified into six categories namely heuristics,metaheuristics, hyperheuristics, mathematical optimization, matheuristics and hybrid approaches. Metaheuristics are the most common choice in addressing NRPs (Chong et al., 2022). Many heuristic approaches were straightforward automation of manual practices, which have been widely studied and documented (Jelinek & Kavois, 1992).

2.3. DEFINITION OF RELATED TERMS

- Roster: A roster is a plan or arrangement detailing the scheduling of nurses to particular shifts, considering numerous requirements and considerations. They may be subject to some constraints.

It is the end product of an attempt to solve the Nurse Rostering Problem. Rosters typically include details such as the planning period (e.g., 28 days), shift types (day shift and night shift), the number of nurses involved, and specific shift timings (e.g., day shift from 07:00 to 15:00 and night shift from 15:00 to 23:00) (Václavík, Šůcha, & Hanzálek, 2016).

- Staffing: Staffing is essentially the allocation of nursing personnel to various shifts within a healthcare facility. It is a very critical component as it directly impacts patient care quality, staff satisfaction, and operational efficiency. Hospitals need to be staffed 24 hours a day over seven days a week (Simić, Simić, Milutinović, & Djordjević, 2014). It is usually based on various factors such as skill levels, qualifications, preferences, and constraints. Effective nurse staffing in hospitals goes beyond simply filling shifts, it's a strategic process that considers multiple factors to ensure optimal patient care, nurse satisfaction, and efficient resource allocation.

- Shifts: These are specific work periods assigned to nurses in a hospital. They refer to the designated periods of time during which nurses are scheduled to work.It coincides with peak patient activity and administrative tasks.There are commonly three different nursing shift schedules; day shift, evening shift and night shift.

(i) Day shifts typically spans the hours of early morning to late afternoon, allowing nurses to work during regular business hours. They vary from 8 to 12 hours, typically running from early morning to late afternoon. This could mean working three 12-hour shifts or four 10-hour shifts per week, depending on the schedule ("Understanding the Different Types of Nursing Shifts | CareerStaff," 2022).

(ii) Evening shifts cover the later part of the day, occuring in the late afternoon or early evening and extending into the night. It is usually an 8-hour shift, Mondays to Fridays between 3-5pm, specific times can vary based on the facility. Nurses on these shifts handle patient care during the day-to-night transition, including admissions, discharges, and ongoing needs.

(iii) Night shifts handle patient care through the night, typically starting around 11 pm and ending around 7 am. There are also 12-hour shifts, which are between 7pm-7am ensure continuous monitoring and care, especially in critical situations.These shifts are crucial in critical care units like intensive care where patients require constant monitoring.

- Planning Period: This refers to the timeframe for which a nurse staffing schedule is created. It refers to the period of time nurses are assigned to specific shifts. It may be daily, weekly, monthly, quarterly, or annually, depending on the specific needs of the organization or project.

- Constraints: Constraints refer to the rules and limitations that must be considered when creating nurse schedules. They ensure that the schedules are feasible, fair, efficient, and and legally compliant. They can abe broadly categorized into two main types; Soft and Hard constraints.

- Scheduling : Scheduling refers to the process of assigning specific shifts to individual nurses over a specified period, typically covering days, weeks, or months, while meeting established constraints and requirements. Its goal is to create effective schedules that ensure adequate coverage, meet operational demands, and take into consideration variables like nurse preferences, skills, and availability in addition to limitations like necessary staffing levels, restrictions on the number of consecutive workdays, and a fair distribution of the workload. Effective scheduling in nurse rostering plays a vital role in ensuring a well-functioning healthcare system by promoting nurse satisfaction and patient safety.

2.4. REVIEW OF RELEVANT CONCEPTS

2.4.1. HYPER-HEURISTICS

According to Cowling et. al(2001), Hyper-Heuristics can be defined as an approach that operates at a higher level of abstraction than metaheuristics and manages the choice of which low-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration. The implementation of Hyper-Heuristics search within a search space of heuristics as opposed to heuristics and Meta- Heuristics that search within a search space of problem solutions. They are, as one can say, a "heuristic which chooses heuristics"(Soubeiga, 2003).

They aim at solving hard computational search by working on a set of heuristics to automate the process of selecting, combining, generating or adapting several simpler heuristics (or components of such heuristics). When using hyperheuristics, we are attempting to find the right method or sequence of heuristics in a given situation rather than trying to solve a problem directly. Instead, it selects at each step of the solution process the most promising simple low-level heuristic (or combination of heuristics) which is potentially able to improve the solution. On the other hand, if there is no improvement, i.e., a locally optimal solution is found, the hyperheuristic diversifies the search to another area of the solution space by selecting appropriate heuristics from the given set (Chakhlevitch & Cowling, 2008). Hyperheuristics aim at reducing the amount of domain knowledge in the search methodology, such that they can function without the need for in-depth understanding of specific low-level heuristics or the intricate workings of the problem's objective function, except for the outcome it produces. Their emphasis is on grasping the direction of the optimization process (maximization or minimization) and evaluating the values of one or more objective functions.(Chakhlevitch & Cowling, 2008).

2.4.2. META-HEURISTICS

Almufti et al.(2023) defined Meta-Heuristics as a higher level of heuristics that function as "master strategies that direct and modify other heuristics to produce solutions beyond those that are typically generated in a quest for local optimality". They attempt to find the best solution out of all possible solutions of an optimization problem.

Meta-heuristics can often find good solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics (Blum & Roli, 2003). They are especially used for solving sophisticated optimization problems like NP-Hard problems. Metaheuristics don't make sure that the optimal solution for a problem is given, they conduct a partial search on the solution space, and the evaluation of the results is based on a set of defined variables or an objective function. They may frequently identify good solutions in combinatorial optimization with less computing work than optimization algorithms, iterative techniques, or basic heuristics since they search through a vast range of viable alternatives. Because of this, they are effective strategies for solving optimization issue (Sadeeq etal., 2021). They are classified into three; Local Search Meta-heuristics(Simulated Annealing, Tabu Search, Viable Neighbourhood search), Constructive Meta-heuristics (Ant Colony Optimization, Large Neighbourhood search) and Population Based Meta-heuristic(Evolutionary Algorithms).

2.4.3. ANT COLONY OPTIMIZATION

First proposed by Dorigo(1992), the Ant Colony Optimization algorithm is an umbrella term for a set of related constructive Meta-Heuristics that imitate the foraging patterns of ants to generate solutions. Artificial ants are used to find optimal solutions to optimization problems. The algorithm is based on how real ants could manage to establish shortest path routes from their colony to feeding sources and back(Dorigo, Maniezzo,

& Colorni, 1996). The domain of application of ACO algorithms is vast. In principle, ACO can be applied to any discrete optimization problem for which some solution construction mechanism can be conceived(Dorigo & Stützle, 2010).

The Ant Colony Algorithm is a pheromone-based algorithm. This means, it is based upon real ants laying down pheromones to lead other ants to the nearest food source, the shorter the path is, the stronger the intensity of the phermone trail, and the more attractive the trail becomes, hence, the shorter and more optimal the path is. The process is characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path (Dorigo, Maniezzo, & Colorni, 1996).

The first ACO algorithm to be proposed was AS(Ant System) by Dorigo(1996), and since then, multiple variations have been developed, like the Ant Colony System(Dorigo & Gambardella, 1997), Elitist Ant System(Sorin et al., 2008), Max-Min Ant System(S-tutzle & Hooz, 2000), Rank-Based Ant System(ASrank), Parallel ant Colony optimization, Continuous Orthogonal Ant Colony, Recursive Ant Colony Optimization(Gupta, Gupta, Arora, & Shankar, 2012). This paper will be focusing on just two of these variations. It will be comparing the performance of the Ant System and the Ant Colony System on the Nurse Rostering Problem.

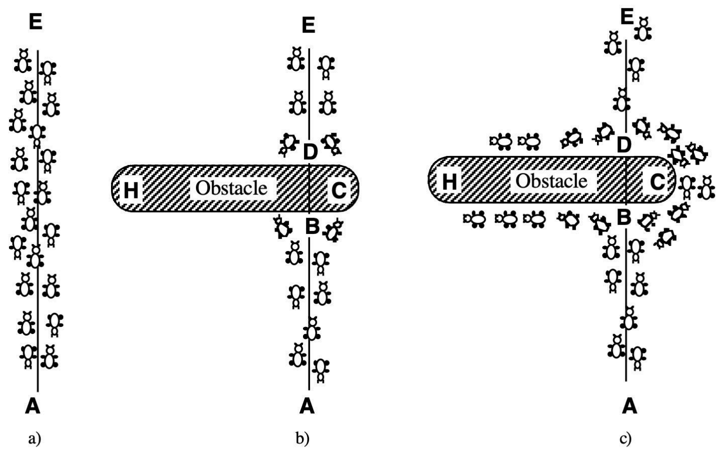

Fig. 1. An example with real ants.

a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

Figure 2.1: Real ants looking for a shorter path when faced with an obstacle.

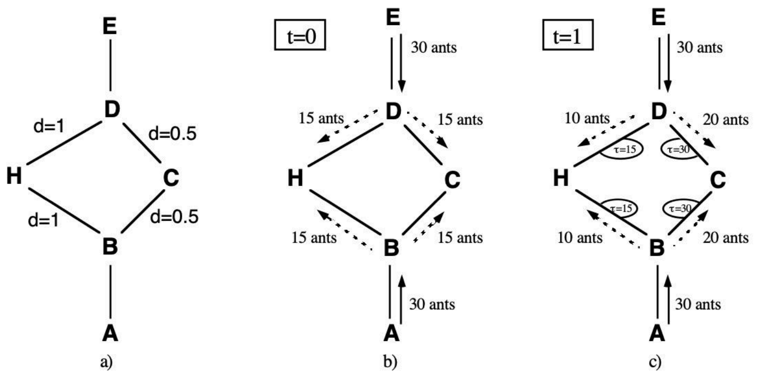

Fig. 2. An example with artificial ants.

a) The initial graph with distances.
b) At time t=0 there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability.
c) At time t=1 trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

Figure 2.2: Same example but with artificial ants.

2.4.4. ANT COLONY SYSTEM

This is a prominent variation of the ACO algorithm family. Proposed by Dorigo and Gambardella(1997), it builds upon the core principles of ACO while introducing specific enhancements to improve performance. It is built upon its predecessor, the ant system, but it differs from it in three main aspects : (i) the state transition rule provides a direct way to balance between exploration of new edges and exploitation of a priori and accumulated knowledge about the problem (ii)the global updating rule is applied only to edges which belong to the best ant tour (iii) while ants construct a solution, a local pheromone-updating rule (local updating rule, for short) is applied to change the phermone level of the edges they are selecting (Dorigo and Gambardella,1997). The role of the local updating rule is to change the attractiveness of edges dynamically, so that when ants use an edge, it becomes slightly less appealing because some of its pheromone is lost. This helps ants utilize pheromone information better. Without local updating, ants would tend to explore only a small area around the best previous tour. The edge selection favours the shortest edges with the largest amount of pheromones, and only the ant that has the shortest tour at the end of each iteration is allowed to update the pheromone.

2.4.5. ANT SYSTEM

This is the first ACO algorithm to be proposed by Dorigo(1996). It is the foundational algorithm within the ACO family. It is the one upon which other variations of ACOs are built upon. It embodies the core principles of ACO like the artificial ants, which are based upon real ants, that explore the solution space of the optimization problem; pheromone trails, which lead the ants towards the more promising paths and helps determine which path is the shortest and more optimal route; stochastic or probablity-based decisions, which are influenced by pheromones and other factors, and the updating of these phermone trails based on the performance of all the ants that complete the tour. Although ant system was useful for discovering good or optimal solutions for small TSP's (up to 30 cities), the time required to find such results made it infeasible for larger problems(Dorigo and Gambardella,1997), this is why improvement on this algorithm was necessary, which gave rise to the variations of ACOs we have today.

2.4.6. HYFLEX

HyFlex, short for Hyper heuristics Flexible Framework, is a JAVA oriented framework for the implementation and comparison of different iterative general-purpose heuristic search algorithms, like Hyper-Heuristics. It features a common software interface for dealing with different combinatorial optimisation problems and provides the algorithm components that are problem specific (4). It provides six hard combinatorial problem domain modules, fully implemented, which include; maximum satisfiability, one dimensional bin packing, permutation flow shop,personnel scheduling, traveling salesman and vehicle routing. The framework allows algorithm designers and implementers to focus on creating practical, general-purpose optimization algorithms without needing in-depth knowledge of the specific problem domains. Researchers in combinatorial optimization often face limitations due to the scarcity of problem domains for testing their adaptive methods. This limitation arises from the significant effort required to implement advanced software components—such as problem models, solution representations, objective function evaluations, and search operators—for various combinatorial optimization problems. The goal of this approach is to enable a hyper-heuristic algorithm, once developed, to be applied to new problems simply by substituting the set of low-level heuristics and the evaluation function.

At the highest level, this framework contains two abstract classes, which are the Problem Domain and the Hyper-Heuristic.

An implementation of the Problem Domain class provides:

i A user-configurable memory or a population of solutions that can be managed by the hyper-heuristic through one or more methods.

ii A routine to randomly initialise solutions, *initializeSolution(i)*, where *i* is the index of the solution array in the memory.

iii A set of problem-specific heuristics that are used to modify solutions. They are called by the *applyHeuristics(i,j,k)* method; where i is the index of the heuristic to call, j is the index of the solution in memory to modify and k is the index in memory where the resulting solution should be placed. There are four groups that each problem-specific heuristic in each problem domain can be classified under, and these are:

- Mutational Heuristics: These heuristics perform modifications to a solution by making changes to its components, like a swap, change, removal or an addition. Examples: Swap Mutation, Gaussian Mutation, Insertion Mutation
- Ruin-recreate Heuristics: These heuristics involve partially "ruining" or destroying a solution and then "recreating" or repairing it to explore new regions of the solution space and potentially find better solutions. They can be broken down into two phases: the ruin phase and the recreate phase. They are, however, considered as large neighbourhood structures and incorporate problem-soecific construction heuristics to rebuild the solutions.
- Local Search Heuristics: These heuristics are optimization techniques that are used to iteratively improve a solution by making small, local changes. They are commonly applied to combinatorial optimization problems where the goal is to find the best solution from a finite set of possible solutions. hey incorporate an iterative improve-ment process and they guarantee that a non-deteriorating solution will be produced.
- Crossover Heuristics: These heuristics combine the genetic information of two or more parent solutions to produce new offspring solutions. The goal is to combine beneficial traits from different parents to explore new areas of the solution space and potentially create better solutions.

iv A varied set of instances that can be loaded using the *load instance* method.
v A fitness function.

The HyFlex framework, renowned for its role in the Cross-domain Heuristic Search Challenge (CHeSC) 2011, stands as a standard benchmark for evaluating cross-domain hyper-heuristics. Offering a common interface for implementing and comparing hyper-heuristics across diverse problem domains, it simplifies the implementation of ant colony hyper-heuristic schemes by abstracting low-level heuristics and providing a standardized performance metric. With its established reputation in the research community, using HyFlex for the implementation of my ant colony schemes ensures efficient development and evaluation.

2.5. MODEL DESCRIPTION

There is an already existing framework that contains a data file format for which each instance can select a combination of objectives and constraints from a wider choice. It has all the functions for this constraints. It also consists methods for parsing these data files, data structures(which can be used by heuristic algorithms) and libraries for visualization of instances and solutions. This software framewor Hyper-Heuristic Flexible Framework(HyFlex), has included a model. The constraints in the original problem formulation has been changed to objectives, and the only constraint is that each employee can only work one shift per day.

The parameters include:

I = Set of nurses available.

$I_t \mid t \in \{1,2,3\}$= Subset of nurses that work 20, 32, 36 hours per week respectively, $I = I_1 + I_2 + I_3$.

J = Set of indices of the last day of each week within the scheduling period = {7, 14, 21, 28, 35}.

K = Set of shift types = {1(early), 2(day), 3(late), 4(night)}.

K’ = Set of undesirable shift type successions = {(2,1), (3,1), (3,2), (1,4), (4,1), (4,2), (4,3)}.

d_{jk} =Coverage requirement of shift type k on day j

m_i = Maximum number of working days for nurse i.

n_1 = Maximum number of consecutive night shifts.

n_2 = Maximum number of consecutive working days.

c_k = Desirable upper bound of consecutive assignments of shift type k.

g_t = Desirable upper bound of weekly working days for the t-th subset of nurses.

h_t = Desirable lower bound of weekly working days for the t-th subset of nurses.

Decision variables:

x_{ijk} = 1 if nurse i is assigned shift type k for day j, 0 otherwise

The constraints are:

i A nurse may not cover more than one shift each day.

The objectives are formulated as (weighted w_i) goals. The overall objective function is:

$$\text{Min } \bar{G}(\text{x}) = \sum_{i=1}^{16} w_i \bar{g}_i(\text{x})$$

Where the goals are:

i Complete weekends (i.e. Saturday and Sunday are both working days or both off).

ii Minimum of two consecutive non-working days.

iii A minimum number of days off after a series of shifts.

iv A maximum number of consecutive shifts of type early and late.

v A minimum number of consecutive shifts of type early and late.

vi A maximum and minimum number of working days per week.

vii maximum of three consecutive working days for part time nurses.

viii Avoiding certain shift successions (e.g. an early shift after a day shift).

ix Maximum number of working days.

x Maximum of three working weekends.

xi Maximum of three night shifts.

xii A minimum of two consecutive night shifts.

xiii A minimum of two days off after a series of consecutive night shifts. This is equivalent to avoiding the pattern: night shift – day off – day on.

xiv Maximum number of consecutive night shifts.

xv Maximum number of consecutive working days.

xvi Shift cover requirements.

2.6. REVIEW OF RELATED METHODS

Based on existing literature, there are several other methods and algorithms that have been employed to solve the Nurse Rostering Problem. Some of these methods include; Simulated Annealing, Tabu Search, Genetic Algorithm, and some other variations of the Ant Colony algorithm.

2.6.1. ANT COLONY OPTIMIZATION ALGORITHMS

Ant Colony algorithms seek to find optimal solutions to optimization problems by imitating the pattern ants use to search for food from their colonies. Over the years, multiple variations of this algorithms have been implemented to solve the Nurse Rostering problem, each working to cater for the shortcomings of the other.

Bunton, Ernst, and Krishnamoorthy (2017) proposed an ACO algorithm using an integer programming based solution construction method to ensure feasibility and select from a collection of schedules, because the NRP is heavily constrained, finding feasible solutions for it can prove difficult, which can result in a poor performance for ACOs. Their approach also used a novel solution merging step that combines the information from multiple ants to generate a better final roster. The only limitation stated in their work was that while using this approach, they were unable to compete against an integer programming software (Gurobi) in solution quality in small to medium sized instances, but despite that, they were able to outperform Gurobi and other methods in large instances.

Achmad, Wibowo, and Diana (2021) proposed an improvement on the Ant Colony algorithm with semi-random initialization for Nurse Rostering Problem. The semi-random initialization took care of the hard contraints, improving the constrution solution phase by assigning nurses directly to the shift related to the hard constraints inorder to avoid any violation of the hard constraints from the get-go, then the rest of the scheduling process will be handled using the ant colony optimization process. This approach was found to have a minimal objective value than the native ACO algorithm, also, the adoption of the semi-random initialization concept was successful in reducing the violation of hard constraints and improving the solution constructed with ant colony, since a violation of hard constraints make a solution not feasible. Unfortunately, this method can only perform optimally with small to medium dimension data.

Some researchers have even gone into hybridization of the algorithm, like Ramli, Abdul Rahman, and Rohim (2019) that used a hybrid Ant Colony Optimization algorithm along with a hill-climbing technique for solving NRPs, which was aimed at fine-tuning the initial solution or roster generated by the ACO algorithm to achieve better rosters, it is proposed to do this in three phases; generating an initial roster, using the ACO to update the roster and implement the hill climbing to look for a more refined solution. However, the solution is mostly optimal with a larger value of pheromones.

Jaradat et al. (2019) demonstrated the use of the hybrid Elitist Ant System on the Nurse Rostering Problem which employed an external memory comprising a set of solutions that assists in the preservation of balance between diversity and quality of the search. The method was found to obtain outstanding and optimal results, using an explicit memory particularly offers diverse high-quality solutions from which the proposed hybrid Elitist-AS can initiate its search process to obtain better solutions. The usage of ACO algorithms for nurse rostering problems has been explored, but not extensively, which is quite worrisome, given that ACOs have advantages over other metaheuristics for solving NRPs such that: can effectively handle the complex constraints often present in nurse rostering; they can be designed to consider multiple objectives simultaneously, for multi-objective NRPs; its dynamic nature, as a result of its phermone updating characteristics allows it to adapt to schedule changes or new constraints that arise and ACOs can be tailored to promote equitable shift assignments among nurses, unlike some metaheuristics that might prioritize efficiency over fairness.

2.6.2. SIMULATED ANNEALING

Simulated annealing is a stochastic local search algorithm for solving unconstrained and bound-constrained optimization problems. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy. (“What Is Simulated Annealing? - MATLAB & Simulink,” n.d.). It draws its inspiration from the metallurgical concept of annealing, where a material undergoes heating and gradual cooling to eliminate impurities.

In this algorithm, an initial temperature, typically set to 1, and a minimum temperature (around 10^{-4}) are defined. The temperature decreases exponentially until it reaches the minimum. At each temperature, the optimization routine runs multiple times, involving the exploration of neighboring solutions and accepting them probabilistically based on energy differences. This stochastic approach allows the algorithm to avoid local minima and improve the chances of converging towards the global optimum. Simulated Annealing proves particularly effective in large, discrete search spaces prioritizing optimality over computational time. (“Simulated Annealing - GeeksforGeeks,” 2017; “What Is Simulated Annealing?,” n.d.). It can be aplied to a range of optimization problems, including NP-hard problems.

A number of people have used this algorithm for nurse rostering problems like Ceschia, Guido, and Schaerf (2020), that used a local search method based on a large neighborhood to solve the static version of the problem defined for the Second International Nurse Rostering Competition (INRC-II). The search method, driven by a simulated annealing metaheuristic, used a combination of neighborhoods that either change the assignments of a nurse or swap the assignments of two compatible nurses, for multiple consecutive days. The method was able to improve on all previous approaches on some datasets, and to get close to the best ones in others.

Knust and Xie (2017) implemented two solution methods, including a MIP (Mixed Integer Programming) model to compute good bounds for the test instances and a heuristic method using the simulated annealing algorithm for practical use. Both methods were tested on the available benchmark instances and on the real-world data. The solution methods in this work were integrated into Vivendi PEP(a staff scheduling software provided by Context Communication GmbH). The approach was found to be able to solve most of the benchmarking instances optimally, proving that the mathematical model accurately describes the benchmarking instances and finds good solutions for the test instances with the MIP model, with the simulated annealing heuristic approach, it showed that it produces good results for both the benchmarking and the test instances in a short amount of time, and it also proves that it is robust concerning different problem instances. However, it did not adapt well to a fixed given running time. On one hand, if the user wants to wait longer to get a better solution, it does not use the additional time efficiently to further enhance the quality of the solution. On the other hand, it does not adapt to situations where a solution must be produced in a very short amount of time.

Furthermore, Ayan-Koç & Aktan, (2019), aimed at creating the best work schedule for 15 nurses working in 2 shifts at a hospital operating 24 hours a day by using Simulated Annealing (SA) Algorithm and developed a new temperature reduction technique and a new assignment technique for the annealing algorithm, which was called "multiple simulated annealing with double assignment". It was found to be quickest when considering that the number of constraints and the number of days in the scheduling problem discussed were quite little, but might have issues generating for larger instances. Simulated annealing is quite wide-spread when it comes to solving problems, esprcially complex ones like Traveling Salesman problem, knapsack problem and so on, but it has several disadvantages, as opposed to ACOs, when applied to nurse rostering problems, including slow convergence, sensitivity to parameter choices, lack of guarantees, lack of collaboration, memory requirements, and lack of exploitation (Knust and Xie, 2017).

2.6.3. TABU SEARCH

Originally proposed by Glover in 1986, Tabu search is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality, it is based on the premise that problem solving, in order to qualify as intelligent, must incorporate adaptive memory and responsive exploration (Glover & Laguna, 1997). It is an adaptive procedure for solving combinatorial optimization problems with the ability to make use of many other methods, like linear programming algorithms and specialized heuristics, which it directs to overcome the limitations of local optimality(Glover, 1989).

Schrack et al. (2021) combined tabu search and genetic algorithm to optimally determine nurse's schedules for smaller scale clinics that require only two nurses per shift and two shifts per day as opposed to the conventional three. They were able to demonstrate that the order in which these meta-heuristic methods are applied can provide better solutions than initially hypothesized. They however plan to improve on the algorithm to cater for larger facilities and to take on additional soft constraints.

Other approaches have been attempted using just Tabu Search like Ramli, Ahmad, Abdul-Rahman, and Wibowo (2020) who explored on the neighbourhoods exploitation of TS to solve the problem inorder to produce a faster and more fitted solution. Their methodology consisted of two phases, the initialization, where they used semi-random initialization to get a good initial solution that does not violate any hard constraint, and the neighbourhood phase for further improving the solution quality using the TS algorithm.Their main aim was to move sample points towards a high-quality solution while avoiding local optima by utilising a calculated force value. The rosters produced were of high quality and without bias.

Tabu Search shares numerous similarities with simulated annealing. In fact, simulated annealing can be viewed as a specific variant of tabu search, wherein a move is designated as tabu with a specified probability. Consequently, the limitations of these two methods are quite comparable.

2.6.4. GENETIC ALGORITHM

Genetic algorithms are adaptive heuristic search algorithms inspired by the principles of natural selection and genetics. They are used to solve optimization and search problems by mimicking the process of natural evolution to improve a population of potential solutions (Kanade, 2023). The genetic algorithm iteratively modifies a population of individual solutions. In each iteration, it selects individuals from the current population as parents and uses them to generate offspring for the subsequent generation. Through successive generations, the population "evolves" towards an optimal solution.

The genetic algorithm is applicable to various optimization challenges that conventional methods may struggle with, such as problems featuring discontinuous, nondifferentiable, stochastic, or highly nonlinear objective functions. Additionally, it can handle mixed integer programming problems, where certain components must be integer-valued (“VisibleBreadcrumbs,” 2019). It begins with an initial population of individuals, typically generated randomly. It then goes through a series of iterations, known as generations or epochs, in which the individuals undergo operations such as selection: to choose which routes will be part of the next generation; selection methods aim to favour fitter individuals,which in this case are routes with lower evaluation rates; crossover: to create new routes by combining genetic materials from two parent routes and mutation: to explore new possibilities and avoid getting stuck in local optima. The selection, crossover, and mutation process is continued for a fixed number of generations or until a termination criterion is met (Kanade, 2023).

In recent years genetic algorithms have emerged as a useful tool for the heuristic solution of complex discrete optimisation problems like the nurse rostering problem. They have been applied to develop weekly schedules for wards of up to 30 nurses, taking into account working contracts, nurse preferences, and hard and soft constraints. However, classical genetic algorithm paradigm is not well equipped to handle constraints and successful implementations and usually needs some form of modification (Aickelin, 1998).

OYELEYE et al. (2020) developed a modified Genetic Algorithm based upon the standard GA to get a more optimal solution to the Nurse Rostering Problrm in LAUTECH Teching hospital, Ogbomosho, Oyo State.

Moreover, Amindoust, Asadpour, and Shirmohammadi (2021) proposed a hybrid genetic algo model for the NRP, taking the fatigue of nurses into consideration and providing a 3-shift schedule based on it. They used a real case study, real case study, a department in one of the hospitals in Esfahan, Iran, where COVID-19 patients were hospitalized to validate the efficiency of the proposed model. The model was able to significantly reduce the time it took to generate schedules, and distribute the nurses within different shifts fairly, considering the least fatigue.

However, it’s worth noting that while genetic algorithms offer promising solutions for complex optimization problems like Nurse Rostering Problems (NRPs), they are not without limitations. These algorithms may not always reach optimal solutions, particularly when dealing with highly constrained Nurse Rostering Problems (NRPs). This can result in suboptimal solutions that fail to meet all specified constraints. Additionally, genetic algorithms are highly sensitive to parameter settings such as crossover and mutation rates, population size, and selection criteria. Inaccurate or inappropriate parameter choices can greatly affect the quality of results obtained. Moreover, the algorithm’s applicability in real-world scenarios is further limited by the requirement for significant processing resources to evaluate fitness functions for large-scale NRPs with enormous rosters.

2.7. SUMMARY OF LITERATURE REVIEW

A comprehensive review of the literature reveals a surprising lack of recent and up--to-datrresearch applying Ant Colony Optimization (ACO) algorithms to nurse rostering problems. Even more scarce are comparative analyses evaluating different ACO hyper-heuristic schemes. This paper aims to bridge this gap by providing data-driven insights by comparing the performance of two selected ACO approaches to determine which is more suitable for tackling nurse rostering problems.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1. PREAMBLE

This section details the analysis and design of the proposed system. It presents the requirements for the system, the techniques taken to achieve the system as well as diagrams to model the proposed system.

3.2. THE PROPOSED SYSTEM

The proposed system is an intelligent tool that makes use of two HyperHeuristic schemes each utilizing two different variations of the Ant Colony optimization algorithm. This tool will, after a number of iterations, each generate possible solutions in the form of rosters for the Nurse Scheduling or Rostering problem. The main aim for this tool is the comparison of the implementation two Ant Colony HyperHeuristic schemes, so there will be little to no user involvement with this system.

3.3. THE NURSE ROSTERING PROBLEM

The nurse rostering or nurse scheduling problem entails creating staff schedules called rosters that assign shifts to nurses over a given planning horizon, which is usaully a week or a month while considering various constraints, objectives and goals. These constraints are classified into hard and soft constraints; hard constraints being mandatory rules that must be strictly adhered to as violating them will render a solution not feasible. Examples include: ensuring a required number of nurses for each shift, required skill sets, legal and contractual regulations, and so forth, while soft constraints are preferences that can be bent, they are desirable but not mandatory, violating them just incurs a penalty, examples include: nurses preferred shifts or days off, reducing fatigue by limiting the number of consecutive shifts, ensuring fairness in shift assignments. The main goal when trying to solve this problem is to create a valid roster that adheres to all hard constraints. The nurse rostering problem can be represented using a mathematical model which consists of a set of nurses, shifts and shift types, decision variable, which is a binary variable represented by x_{ijk}, indicating if nurse *i* is assigned to shift *j* on day *k*, with the value of 1 if assigned and 0 if otherwise, the objective function which aims to minimize the overall penalty incurred by violating soft constraints or not meeting desired goals and maximize the nurse's preferences:

$\min \sum_{i,j,k} C_{\mathrm{ijk}} * x_{\mathrm{ijk}} + \sum$ penalties for soft constraints violations

And a bunch of soft and hard constraints that should be met based on the objective function and specific hospital needs. This is just an abstracted explanation of the NRP model, it can be modified and adjusted based on the specific requirements; appropriate constraints and objectives, of the nurse rostering problem being solved.

The Nurse Rostering Problem is usually solved by heuristics and Metaheuristics like Simulated Annealing, Tabu Search, Ant Colony Optimization, and others, mainly because they can generate good solutions in reasonable time. The process of solving this problem with heuristics is broken down into steps which begins with all the data concerning the specific NRP to be solved being gathered and prepared first. Then an iterative process which entails an initial roster being generated randomly or based on a simple rule, which is then evaluated based on the defined objective function (might involve calculating the total penalty incurred by violating soft constraints or not meeting desired goals), this roster is then looked over to identify areas for improvement, like a violation of a high-priority preference or a very desirable constraint, then, depending on the chosen heuristic, there is an attempt to improve the roster. This might involve swapping shifts between nurses, re-assigning a nurse to a different shift type, or modifying the schedule to respect a high-weight preference. This step continues until a stopping criterion is met, which can either be the algorithm reaching a pre-set maximum number of iterations or the algorithm finding an acceptable solution. A post-processing technique can be applied to the final roster if necessary, to fine-tune and ensure optimality of the solution. Then the roster is evaluated based on the objective function and constraints.

3.4. THE HYFLEX FRAMEWORK

The Hyflex framework (Hyper heuristics Flexible Framework) is a versatile software framework designed to facilitate the development and testing of hyper-heuristics across various problem domains. It isn't focused on solving specific problems itself, but rather provides a powerful platform for developing algorithms that tackle a wide range of combinatorial optimization problems. These are problems where you need to find the best possible arrangement from a finite set of options. This framework has six (6) problem domains which are:

i Maximum Satisfiability(SAT): This is also called Boolean Satisfiability, and it is a fundamental problem in computer science and mathematical logic that involves determining whether there exists an assignment of truth values (true or false) to variables in a Boolean formula such that the entire formula evaluates to true. If there does, the formula is called satisfiable, on the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is unsatisfiable. The goal is to determine if there's a way to make a complex formula true through clever assignment of truth values. This problem is NP-complete.

ii One-dimensional bin packing: This is a classic combinatorial optimization problem that has applications in real-world scenarios like resource allocation and logistics. It involves packing a set of items with varying sizes into a finite number of bins, each with a fixed capacity, in such a way that the number of bins used is minimized. Computationally, the problem is NP-hard.

iii Permutation Flow Shop Scheduling (PFSS): This is a production problem for finding the best sequence of jobs to be processed by machines in order to minimize the given objective function. It deals with efficiently scheduling jobs in a production environment with a specific flow structure. Here, a set of jobs must be processed in the same order on a series of machines. The objective is typically to minimize the total completion time (makespan), although other objectives like minimizing total tardiness or maximizing throughput can also be considered. This is also an NP-hard problem.

iv Travelling Salesman Problem: This is a classic combinatorial optimization problem that seeks the shortest possible route for a salesman to visit a set of n cities exactly once and return to the original city, given a set of citiesor locations and the distances between each pair of them. The goal is to find the shortest possible route that visits each city exactly once and returns to the starting city.

v Vehicle Routing: This is a combinatorial optimization and integer programming problem that seeks the optimal set of routes for a fleet of vehicles to deliver goods to a set of customers. The objective is typically to minimize the total delivery cost, which may include distance, time, or other metrics, while satisfying various constraints. It asks the question: What is the optimal set of routes for the vehicles to take such that all deliveries are completed, capacity constraints are met and total route cost is minimized. It generalizes the Travelling Salesman Problem and is an NP-hrad problem.

vi Personnel Scheduling: This is a type of combinatorial optimization problem where the goal is to allocate shifts or work hours to a set of employees while satisfying various constraints and optimizing one or more objectives. Essentially, it involves determining the specific times and shifts that each employee should work over a given planning period.

The Personnel Scheduling problem is in actuality, not a problem on its own, but a title for a group of very similar problems. There is a group of problems with very similar structure but different objectives and constraints; these are personnel scheduling problems. This fact makes it challenging to implement a problem domain for personnel scheduling. The framework overcame this challenge by creating a data file format where each instance can select a combination of objectives and constraints from a wide choice, then a software framework containing all the functions for these objectives and constraints was implemented. The framework also contains methods for parsing these data files, data structures and libraries for visualisations of instances and solutions. The objectives for the Nurse Rostering Problem domain provided by this framework can be categirized into two groups:

- Coverage Objectives which aim to make sure that the preferred number of employees, preferrably with skills, are working each shift.
- Employee working objectives which relates to the individual work patterns for each employee. They aim to maximize the employees' satisfaction with their work schedules.

In the Hyflex Framework, it is necessary to implement a method to initialize a new solution, and these are most usually low-level heursitics. There are four categories of heuristics; Mutation, Local Search, Ruin and Recreate and Crossover. Hyflex currently provides 1 mutation heuristic (Mutation Heuristic 1), three crossover heursitics (Crossover Heuristics 1-3), three ruin and recreate heuristics (Ruin and Recreate Heuristics 1-3) and

5 Local Search Heuristics, which are divided into two, Local Search Heuristics 1-3 are the hill-climbers(or hill-descenders, since we are dealing with minimization), that use of neighbourhood operators, and Local Search Heuristics 4 and 5 which are variants of the variable depth search. The solution for all the problems in the personnel scheduling domain is initialized using Local Search Heuristic 3, which is a hill climber that uses the *new* neighbourhood operator that introduces new shifts into or deletes from the initial roster.

```
1.  WHILE there are untried swaps
2.    FOR BlockLength=1 up to MaxBlockLength
3.      FOR each employee (E1) in the roster
4.        FOR each day (D1) in the planning period
5.          FOR each shift type (S1) (including day off)
6.            Remove all assignments for E1 on D1 up to D1+BlockLength
              and assign shifts of type S1 to E1 on D1 up to
              D1+BlockLength
7.            IF an improvement in roster penalty THEN
8.              Break from this loop and move on to the next day (D1+1)
9.            ELSE
10.             Reverse the swap
11.           ENDIF
12.         ENDFOR
13.       ENDFOR
14.     ENDFOR
16.   ENDFOR
17. ENDWHILE
```

Figure 3.3: Pseudo-code for Local Heursitic 3

Table 3.2: The Total Heuristics Provided by Hyflex for each Heuristic Type

Domain	***Total***	***Mut***	***R&R***	***Crossover***	***LS***
Maximum Satisfiability	9	4	1	1	2
Bin Packing	8	3	2	1	2
Permutated Flow Shop	15	5	2	3	4
Personnel Scheduling	12	1	3	3	5
Travelling Salesman	15	5	1	3	6
Vehicle Routing	12	4	2	2	4

3.5. THE ANT SYSTEM

This is a type of algorithm that is a part of a broader class of algorithms known as Ant Colony Optimization (ACO). The main idea is to use artificial ants to simulate the way real ants find the shortest paths between their nest and food sources by laying down pheromones, which guide other ants towards the discovered path. Originally proposed by Dorigo in 1996, it was first applied to the Travelling Salesman Problem and is the progenitor of all further research work on Ant Colony algorithms.

It is a set of algorithms inspired by the foraging behaviour of ants specifically designed for solving combinatorial optimization problems. Real ants find the shortest path to a food source by something called 'Pheromone Trails', this concept is one of the core mechanisms of the Ant System. Ants lay pheromones on the paths they traverse, and the amount of pheromone deposited depends on the quality of the solution, like, shorter paths have much thicker pheromones because more ants would have traversed it. Over time, pheromone trails evaporate, preventing the system from converging too quickly to a suboptimal solution and encouraging exploration. The ants are endowed with a working memory M_i^k, that stores nodes or vertices already visited; at initialization, M_i^k is an empty set, for every ant i$(1 \leq i \leq k)$,where k is the total number of vertices, then as ant i visits vertex v, it is added to memory M_i^k. In the original work by Dorigo,Maniezzo,& Colorni(1996), a tabu list was used as a memory to store the already visited cities. The ants begin at a particular node or location and the choice of the next node to visit is based on a probability that is a function of the intensity of the phermone on the trail(τ_{ij}) between the start node and the destination node and a heuristic value (η_{ij}) which is often based on an inverse of the distance between the two nodes. The formula is usually given as:

$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k \in N_k(i)} [\tau_{ik}(t)]^{\alpha}.[\eta_{ik}]^{\beta}}$

where $N_k(i)$ is the set of nodes that ant *k* has not yet visited, α and β are parameters that control the relative importance of trail versus visibility, so the transition probability $P_{ij}^{k}(t)$ is a trade-off between visibility, which says that closer nodes should have more preference thereby implementing a greedy constructive heuristic and trail intensity, which says that the edge between nodes that has had more traffic is more desirable, thereby implementing the autocatalytic process. After every *n* iterations of the algorithm (which is also called a cycle), the pheromone trail intensity is updated using this formula:

$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$

where:

$(1-\rho)$ represents the evaporation of trail between time t and t+n,

$\Delta\tau_{ij}^{k}$ is the quantity per unit length of pheromones laid on the edge(i,j) by the k-th ant between time t and t+n and is given by:

$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$; if the k-th ant uses edge (i,j) in its tour (between time t and t + n)

where Q is a constant and L_k is the tour length of the k-th ant.

This is called the Global Pheromone Update, where a greater amount of pheromones are allocated to shorter tours. The coefficient ρ must be a value less than 1 to avoid saturation of trail and encourage exploration (if the ρ value is < 1, pheromone values decay gradually overtime, thus reducing the influence of previously deposited pheromones, which allows ants explore new paths instead of only following the most reinforced ones, to avoid premature convergence to suboptimal solutions).

```
1. Initialize:
      Set t:=0                          {t is the time counter}
      Set NC:=0                         {NC is the cycles counter}
      For every edge (i,j) set an initial value $\tau_{ij}(t)=c$ for trail intensity and $\Delta\tau_{ij}= 0$
      Place the m ants on the n nodes

2. Set s:=1                             {s is the tabu list index}
   For k:=1 to m do
      Place the starting town of the k-th ant in $\mathbf{tabu_k}(s)$

3. Repeat until tabu list is full       {this step will be repeated (n-1) times}
      Set s:=s+1
      For k:=1 to m do
         Choose the town j to move to, with probability $p_{ij}^{k}(t)$ given by equation (4)
                                        {at time t the k-th ant is on town i=$\mathbf{tabu_k}$(s-1)}
         Move the k-th ant to the town j
         Insert town j in $\mathbf{tabu_k}(s)$

4. For k:=1 to m do
      Move the k-th ant from $\mathbf{tabu_k}(n)$ to $\mathbf{tabu_k}(1)$
      Compute the length $L_k$ of the tour described by the k-th ant
      Update the shortest tour found
   For every edge (i,j)
      For k:=1 to m do
         $\Delta\tau_{i,j}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if } (i,j) \in \text{tour described by } \mathbf{tabu_k} \\ 0 & \text{otherwise} \end{cases}$
         $\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau_{ij}^{k};$

5. For every edge (i,j) compute $\tau_{ij}(t+n)$ according to equation $\tau_{ij}(t+n)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij}$
   Set t:=t+n
   Set NC:=NC+1
   For every edge (i,j) set $\Delta\tau_{ij}:=0$

6. If ($NC < NC_{MAX}$) and (not stagnation behavior)
      then
         Empty all tabu lists
         Goto step 2
      else
         Print shortest tour
         Stop
```

Figure 3.4: Pseudocode of the ant cycle algorithm in the Ant System for TSP.

3.6. THE ANT COLONY SYSTEM

This is another variation of the Ant Colony Optimization algorithm. It, like its parent algorithm is inspired by the foraging behaviour of ants. The Ant Colony System algorithm is built upon the Ant system, with only a few changes to improve its effectiveness. It was noted by Dorigo &Gambardella(1997) that for larger problems, the time given to the Ant System made infeasible to find good or optimal solutions. It was also discovered that even when given more time, the algorithm still suffered from slow convergence and had the tendency to get stuck in local optima instead of converging to the global optimum. So, the Ant Colony System was developed to accelerate convergence, while still searching the solution space effectively, and also to balance exploration and exploitation inorder to avoid suboptiomal solutions. The Ant Colony System differs from the Ant System in these ways:

i While the Ant System uses a purely probabilistic approach for selection of the next path to be taken by the ant, this algorithm has a state transition rule that ensures there is a way to balance between exploration of new paths and exploitation of already reinforced paths to help avoid convergence of solution to local optima.

ii The Global Pheromone Update rule, which is only applied to the edges that belong to the tour of the best ant, which is the ant that constructed the best solution. In Ant System, however, all the ants contribute to the global pheromone update.

iii The Local Pheromone Update rule, which happens while ants construct a solution, and every ant has completed a tour. This is done only in Ant Colony System, and not in Ant System, to encourage exploration.

iv The Ant System applies evaporation of pheromones uniformly after each iteration, while the Ant Colony System uses a slower pheromone evaporation rate and combines global updates with evaporation to focus on the best solutions.

The

REFERENCES

Duka, E. (2015). NURSE SCHEDULING PROBLEM. European Scientific Journal, 2, 1857–7881. Retrieved from https://eujournal.org/index.php/esj/article/download/6481/6221.

Glass, C. A., & Knight, R. A. (2010). The nurse rostering problem: A critical appraisal of the problem structure. European Journal of Operational Research, 202(2), 379–389. https://doi.org/10.1016/j.ejor.2009.05.046.

Burke, E. K., De Causmaecker, P., Berghe, G. V., & Van Landeghem, H. (2004). The State of the Art of Nurse Rostering. Journal of Scheduling, 7(6), 441–499. https://doi.org/10.1023/b:josh.0000046076.75950.0b

Václavík, R., Šůcha, P., & Hanzálek, Z. (2016). Roster evaluation based on classifiers for the nurse rostering problem. Journal of Heuristics, 22(5), 667–697. https://doi.org/10.1007/s10732-016-9314-9

Jelinek, R. C., & Kavois, J. A. (1992). Nurse staffing and scheduling: past solutions and future directions. PubMed, 3(4), 75–82.

Simić, D., Simić, S., Milutinović, D., & Djordjević, J. (2014). CHALLENGES FOR NURSE ROSTERING PROBLEM AND OPPORTUNITIES IN HOSPITAL LOGISTICS. JOURNAL of MEDICAL INFORMATICS & TECHNOLOGIES, 23(1642-6037).

Understanding the Different Types of Nursing Shifts | CareerStaff. (2022, January 25). Retrieved from CareerStaff Unlimited website: https://www.careerstaff.com/clinician-life-blog/nursing/understanding-different-nursing-shifts/

Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)

Soubeiga, E.: Development and application of hyperheuristics to personnel scheduling. PhD Thesis, Department of Computer Science, University of Nottingham, UK (2003)

Chakhlevitch, K., & Cowling, P. (2008). Hyperheuristics: Recent Developments. In C. Cotta, M. Sevaux, & K. Sörensen (Eds.), Adaptive and Multilevel Metaheuristics (pp. 3–29). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/9783540794387_1

E. K. Burke, G. Kendall, and E. Soubeiga, “A tabu-search hyperheuristic for timetabling and rostering,” Journal of Heuristics, vol. 9, pp. 451-470, 2003.

B. Bilgin, P. De Causmaecker, B. Rossie, and G. V. Berghe, “Local search neighbourhoods for dealing with a novel nurse rostering model,” Annals of Operations Research, vol. 194, pp. 33-57, 2012.

M. J. Brusco and L. W. Jacobs, “Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations,” European Journal of Operational Research, vol. 86, pp. 249-261, 1995.

W. J. Gutjahr and M. S. Rauner, “An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria,” Computers & Operations Research, vol. 34, pp. 642-666, 2007.

B. Maenhout and M. Vanhoucke, “Branching strategies in a branchand-price approach for a multiple objective nurse scheduling problem,” Journal of Scheduling, vol. 13, pp. 77-93, 2010.

M. A. Awadallah, A. T. Khader, M. A. Al-Betar, and A. L. a. Bolaji, “Global best Harmony Search with a new pitch adjustment designed for Nurse Rostering,” Journal of King Saud University-Computer and Information Sciences, vol. 25, pp. 145-162, 2013.

M. A. Awadallah, A. T. Khader, M. A. Al-Betar, and A. L. a. Bolaji, “Harmony search with greedy shuffle for nurse rostering,” International Journal of Natural Computing Research (IJNCR), vol. 3, pp. 22-42, 2012

E. K. Burke, T. Curtois, R. Qu, and G. V. Berghe, “A scatter search methodology for the nurse rostering problem,” Journal of the Operational Research Society, vol. 61, pp. 1667-1679, 2010

B. Cheang, H. Li, A. Lim, and B. Rodrigues, “Nurse rostering problems––a bibliographic survey,” European Journal of Operational Research, vol. 151, pp. 447-460, 2003.

J.H. Oldenkamp, Quality in Fives: On the Analysis, Operationalization and Application of Nursing Schedule Quality, University of Groningen, 1996.

Almufti, S. M., Awaz Ahmad Shaban, Rasan Ismael Ali, & Dela, J. A. (2023). Overview of Metaheuristic Algorithms. Polaris Global Journal of Scholarly Research and Trends, 2(2), 10–32. https://doi.org/10.58429/pgjsrt.v2n2a144

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization. ACM Computing Surveys, 35(3), 268–308. https://doi.org/10.1145/937503.937505

Sadeeq, H. T., Abdulazeez, A. M., Kako, N. A., Zebari, D. A., & Zeebaree, D. Q. (2021). A New Hybrid Methodfor Global Optimization Based on the Bird Mating Optimizer and the Differential Evolution. 2021 7thInternational Engineering Conference “Research & Innovation amid Global Pandemic" (IEC), 54–60.https://doi.org/10.1109/IEC52205.2021.9476147

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 26(1), 29–41. https://doi.org/10.1109/3477.484436

Dorigo, M., & Stützle, T. (2010). Ant Colony Optimization: Overview and Recent Advances. Handbook of Metaheuristics, (781-3794), 227–263. https://doi.org/10.1007/978-1-4419-1665-5_8

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1), 53–66. https://doi.org/10.1109/4235.585892

T. Stützle et H.H. Hoos, MAX MIN Ant System, Future Generation Computer Systems, volume 16, pages 889-914, 2000

Gupta, D. K., Gupta, J. P., Arora, Y., & Shankar, U. (2012). Recursive ant colony optimization: a new technique for the estimation of function parameters from geophysical field data. Near Surface Geophysics, 11(3), 325–340. https://doi.org/10.3997/1873-0604.2012062

Bunton, J. D., Ernst, A. T., & Krishnamoorthy, M. (2017). An Integer Programming based Ant Colony Optimisation Method for Nurse Rostering. Computer Science and Information Systems (FedCSIS), 2019 Federated Conference On, 11(2300-5963). https://doi.org/10.15439/2017f237

Achmad, S., Wibowo, A., & Diana, D. (2021). Ant colony optimization with semi random initialization for nurse rostering problem. International Journal for Simulation and Multidisciplinary Design Optimization, 12, 31–31. https://doi.org/10.1051/smdo/2021030

Ramli, R., Abdul Rahman, R., & Rohim, N. (2019). A hybrid ant colony optimization algorithm for solving a highly constrained nurse rostering problem. Journal of Information and Communication Technology, 18(3), 305-326.

Jaradat, G. M., Al-Badareen, A., Ayob, M., Al-Smadi, M., Al-Marashdeh, I., Ash-Shuqran, M., & Al-Odat, E. (2019). Hybrid Elitist-Ant System for Nurse-Rostering Problem. Journal of King Saud University - Computer and Information Sciences, 31(3), 378–384. https://doi.org/10.1016/j.jksuci.2018.02.009

What Is Simulated Annealing? - MATLAB & Simulink. (n.d.). Retrieved from www.mathworks.com website: https://www.mathworks.com/help/gads/what-is-simulated-annealing.html

Simulated Annealing - GeeksforGeeks. (2017, August 11). Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/simulated-annealing/

What is Simulated Annealing? (n.d.). Retrieved from www.cs.cmu.edu website: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html

Ceschia, S., Guido, R., & Schaerf, A. (2020). Solving the static INRC-II nurse rostering problem by simulated annealing based on large neighborhoods. Annals of Operations Research, 288(1), 95–113. https://doi.org/10.1007/s10479-020-03527-6

Knust, F., & Xie, L. (2017). Simulated annealing approach to nurse rostering benchmark and real-world instances. Annals of Operations Research, 272(1-2), 187–216. https://doi.org/10.1007/s10479-017-2546-8

Ayan-Koç, B., & Aktan, M. (2019). The Solution of Nurse Scheduling Problem with Simulated Annealing Algorithm. Journal of Scientific and Engineering Research, 6(4), 153–160. www.jsaer.com

Glover, F., & Laguna, M. (1997). Tabu Search. Boston, MA: Springer US. https://doi.org/10.1007/978-1-4615-6089-0

Glover, F. (1989). Tabu Search—Part I. ORSA Journal on Computing, 1(3), 190–206. https://doi.org/10.1287/ijoc.1.3.190

Schrack, J., Ortega, R., Dabu, K., Truong, D., Aibin, M., & Aibin, A. (2021, September 1). Combining Tabu Search and Genetic Algorithm to Determine Optimal Nurse Schedules. https://doi.org/10.1109/CCECE53047.2021.9569111

Ramli, R., Ahmad, S. N. I., Abdul-Rahman, S., & Wibowo, A. (2020). A tabu search approach with embedded nurse preferences for solving nurse rostering problem. International Journal for Simulation and Multidisciplinary Design Optimization, 11, 10. https://doi.org/10.1051/smdo/2020002

VisibleBreadcrumbs. (2019). Retrieved from Mathworks.com website: https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

Kanade, V. (2023, September 6). Genetic Algorithms - Meaning, Working, and Applications. Retrieved from Spiceworks website: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-are-genetic-algorithms/

Aickelin, U. (1998). Nurse Rostering with Genetic Algorithms.ArXiv (Cornell University).

OYELEYE, C. A., OLADELE, G. O., ALADE, O. M., BELLO, O. A., ADEYEMO, A. I., ABIODUN, E., & ADEDEJI, T. O. (2020). MODIFIED GENETIC ALGORITHM FOR SOLVING NURSE SCHEDULING PROBLEM. International Research Journal of Computer Science, 07(04), 33–41. https://doi.org/10.26562/irjcs.2020.v0704.002

Amindoust, A., Asadpour, M., & Shirmohammadi, S. (2021). A Hybrid Genetic Algorithm for Nurse Scheduling Problem considering the Fatigue Factor. Journal of Healthcare Engineering, 2021, 1–11. https://doi.org/10.1155/2021/5563651

Hakim, L., Bakhtiar, T., & Jaharuddin. (2017). The nurse scheduling problem: a goal programming and nonlinear optimization approaches. IOP Conference Series Materials Science and Engineering, 166, 012024. https://doi.org/10.1088/1757899X/166/1/012024

Nasiri, M. M., & Rahvar, M. (2017). A two-step multi-objective mathematical model for nurse scheduling problem considering nurse preferences and consecutive shifts. International Journal of Services and Operations Management, 27(1), 83. https://doi.org/10.1504/ijsom.2017.083338

(4) (PDF) HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search. Available from: https://www.researchgate.net/publication/258836405_HyFlex_A_Benchmark_Framework_for_Cross-Domain_Heuristic_Search [accessed May 24 2024].

REFERENCES