**[YOUR TOPIC – in caps]**

**BY**

**[SURNAME],** [FIRST-NAME] [MIDDLE-NAME] **([MATRIC. NUMBER])**

**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.**

**JULY, [YEAR]**

## CERTIFICATION

I hereby certify that this project was carried out by [First-Name] [Middle-Name] [Surname-In Caps] in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ogun State, Nigeria, under my supervision.

|  |  |
| --- | --- |
| **Dr. Aderonke A. Oni** |  |
| *Supervisor* | **Signature and Date** |
|  |  |
| **Prof. Olufunke O. Oladipupo** |  |
| *Head of Department* | **Signature and Date** |

## DEDICATION

The dedication section is a brief and personal acknowledgement. It typically includes gratitude to family members for their unwavering support, friends for their understanding, and academic advisors or professors for their guidance and expertise. It might also mention anyone who inspired or motivated the student throughout their academic journey.

## ACKNOWLEDGEMENTS

The acknowledgements section of the project is where you express your gratitude to those who have helped and supported you throughout your project.

## TABLE OF CONTENTS

**CONTENT PAGES**

**COVER PAGE** i

[**CERTIFICATION** ii](#_Toc169632526)

[**DEDICATION** iii](#_Toc169632527)

[**ACKNOWLEDGEMENTS** iv](#_Toc169632528)

[**TABLE OF CONTENTS** v](#_Toc169632529)

[**LIST OF FIGURES** vii](#_Toc169632530)

[**LIST OF TABLES** viii](#_Toc169632531)

[**ABBREVIATIONS** ix](#_Toc169632532)

[**ABSTRACT** x](#_Toc169632533)

[CHAPTER One: INTRODUCTION 1](#_Toc169632534)

[1.1Fundamentals 1](#_Toc169632535)

[1.2Applying a Paragraph/Object Reference 2](#_Toc169632536)

[1.3Figure and Table Captions 3](#_Toc169632537)

[1.4The use of Roman numerals for itemisation 5](#_Toc169632538)

[1.5Inserting Equations 5](#_Toc169632539)

[CHAPTER Two: LITERATURE REVIEW 7](#_Toc169632540)

[2.1Preamble – A sample Heading 2 and Citations 7](#_Toc169632541)

[2.2Other citation styles 7](#_Toc169632542)

[2.2.1 A Heading 3 style 8](#_Toc169632543)

[2.2.2 Another Heading 3 style 8](#_Toc169632544)

[2.3Automated software engineering 8](#_Toc169632545)

[2.3.1 Automated Code Generation 8](#_Toc169632546)

[2.3.2 Automated Testing – This has two Heading 4 sections 9](#_Toc169632547)

[2.3.3 Artificial Intelligence in Software Engineering – This has three Heading 4 sections 9](#_Toc169632548)

[2.4Summary 11](#_Toc169632549)

[CHAPTER Three: METHODOLOGY/SYSTEM ANALYSIS AND DESIGN 12](#_Toc169632550)

[3.1Presenting Algorithms 12](#_Toc169632551)

[3.2More Equations 13](#_Toc169632552)

[3.2.1 Routing Problem I: Travelling Salesman Problem 13](#_Toc169632553)

[3.2.2 Routing Problem Domain II: Vehicle Routing 14](#_Toc169632554)

[3.3Evaluation Schemes 14](#_Toc169632555)

[3.3.1 Testing New Schemes 14](#_Toc169632556)

[3.3.2 Evaluation Procedures 15](#_Toc169632557)

[3.4Summary of Styles used in this template 18](#_Toc169632558)

[3.4.1 Heading 1 18](#_Toc169632559)

[3.4.2 Heading 2 18](#_Toc169632560)

[3.4.3 Heading 3 19](#_Toc169632561)

[3.4.4 Heading 4 19](#_Toc169632562)

[3.4.5 Heading 5 19](#_Toc169632563)

[3.4.6 Numbered-Item & Item 19](#_Toc169632564)

[3.4.7 Caption 21](#_Toc169632565)

[3.4.8 Table-Caption 21](#_Toc169632566)

[3.4.9 Eqn-Caption 21](#_Toc169632567)

[3.4.10 Text 21](#_Toc169632568)

[3.4.11 Text-Narrow 21](#_Toc169632569)

[3.4.12 Table Header & Table-Text 21](#_Toc169632570)

[CHAPTER Four: SYSTEM IMPLEMENTATION AND EVALUATION 22](#_Toc169632571)

[4.1Preamble 22](#_Toc169632572)

[4.2Certification Page 22](#_Toc169632573)

[4.3Dedication Page 22](#_Toc169632574)

[4.4Acknowledgements Page 22](#_Toc169632575)

[4.5Table of Contents Page 22](#_Toc169632576)

[4.6List of Figures Page 23](#_Toc169632577)

[4.7List of Tables Page 24](#_Toc169632578)

[4.8Abbreviations 24](#_Toc169632579)

[4.9Abstract 24](#_Toc169632580)

[CHAPTER Five: CONCLUSION AND RECOMMENDATIONS 25](#_Toc169632581)

[5.1Chapter One 25](#_Toc169632582)

[5.2Chapter Two 25](#_Toc169632583)

[5.3Chapter Three 26](#_Toc169632584)

[5.4Chapter Four 27](#_Toc169632585)

[5.5Chapter Five 28](#_Toc169632586)

[5.6](#_Toc169632587)[References 29](#_Toc169632587)

[**REFERENCES** 30](#_Toc169632588)

[**APPENDICES [Delete this if not applicable to your report]** 32](#_Toc169632589)

[**APPENDIX A: Best Sequences Evolved by Our Scheme** 32](#_Toc169632590)

[**APPENDIX B: A C++ Implementation of the sorting algorithm for random keys** 33](#_Toc169632591)

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURES** | **TITLES OF FIGURES** | **PAGES** |
| 1.1 | [Caption of Figure 1.1] | 3 |
| 1.2 | [Caption of Figure 1.2] | 4 |
| 2.1 | [Caption of Figure 2.1] | 10 |
| 2.2 | [Caption of Figure 2.2] | 12 |
| 3.1 | [Caption of Figure 3.1] | 72 |
| 3.2 | [Caption of Figure 3.2] | 78 |
| 4.1 | [Caption of Figure 4.1] | 122 |
| 4.2 | [Caption of Figure 4.2] | 123 |
| 4.3 | [Caption of Figure 4.3] | 123 |
| A1 | [Caption of the first figure in the appendix section] | 169 |
| A2 | [Caption of the second figure in the appendix section] | 170 |
| A3 | [Caption of the third figure in the appendix section] | 170 |

## LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLES** | **TITLES OF TABLES** | **PAGES** |
| 1.1 | [Caption of Table 1.1] | 4 |
| 2.1 | [Caption of Table 2.1] | 59 |
| 2.2 | [Caption of Table 2.2] | 61 |
| 2.3 | [Caption of Table 2.3] | 68 |
| 3.1 | [Caption of Table 3.1] | 73 |
| 3.2 | [Caption of Table 3.2] | 74 |
| 4.1 | [Caption of Table 4.1] | 158 |
| 4.2 | [Caption of Table 4.2] | 165 |
| 4.3 | [Caption of Table 4.3] | 166 |

## ABBREVIATIONS

|  |
| --- |
| ACO: Ant Colony Optimisation |
| API: Application Programming Interface |
| BERT: Bidirectional Encoder Representations from Transformers |
| CNN: Convolutional Neural Network |
| EA: Evolutionary Algorithm |
| GA: Genetic Algorithm |
| LSTM: Long Short-Term Memory |
| NLP: Natural Language Processing |
| RL: Reinforcement Learning |
| SIFT: Scale-Invariant Feature Transform |
| YOLO: You Only Look Once  RCWYA: Replace Contents With Your Abbreviations |

## ABSTRACT

Computationally intensive tasks, such as large combinatorial optimisation problems, have been addressed over the years using advanced search techniques, with genetic algorithms among the most prominent. Genetic algorithms are evolutionary techniques that simulate natural selection to generate high-quality solutions for challenging combinatorial optimisation problems (COPs).

[Use the **Text-Narrow** style for the Abstract; it should not be more than one page]

# 

**INTRODUCTION**

## 1.1 Fundamentals

This template has been carefully designed to protect students from submitting their reports with diverse formatting styles. The student should follow the style instructions strictly to avoid messing up the formatting tasks.

To use a formatting style. It is recommended that the student follow either of the three steps below:

1. Copy and paste the text into Notepad, then copy and paste from the Notepad into the text section with the already applied formatting style.
2. Copy the text from the source document and then avoid pasting the text directly; instead, right-click on the area the student wants to paste his/her text intro and paste with the "Keep Text Only" option.
3. The third option applies when the student uses the template to write his/her report from scratch. Type the text, highlight the text and select the formatting styles the students desire to apply from the Styles ribbon on the Home tab.

There are three main headings (**Heading 2 - 2.3, Heading 3 - 2.3.3, and Heading 4 - A**). Avoid going past Heading Four to reference sections within the report easily. Heading 2 was applied to the text above (1.1. Background Information).

**The Heading 1 style**: This is applied only to the Chapter headings (Chapter One, Chapter Two, … Chapter Five). Since the report is in five chapters, do not edit the styles; only change the text of the chapter's title (if necessary). Read and follow the instructions in the remaining part of the template to see other report format instructions concerning other aspects of report writing.

**The Text Style**: Most of the text in the report should use this style. This text is based on this style. Its main features are Times New Roman, 12 pt, and 1.5 line spacing.

**The Item Style**: It is a Roman numeral style for itemisation that comes in handy for the aim and objectives section and any other part of the report that requires students to itemise contents.

## 1.2 Applying a Paragraph/Object Reference

A paragraph is a distinct section of writing that focuses on a single idea or theme. It typically consists of several sentences that develop and support the main point. This section has four paragraphs: this current one and another three to demonstrate what it should look like. Applying a paragraph in this format is simple; in the last line of the previous paragraph, instead of pressing the "Enter" key unnecessarily, set the "After" spacing of the paragraph to 6pt (a depiction of this can be seen in Figure 1.1) in the paragraph settings of the paragraph section of perhaps, the Home tab. Every table and figure in the report should be referenced before they appear. The student should notice that this principle was applied to Figure 1.1.

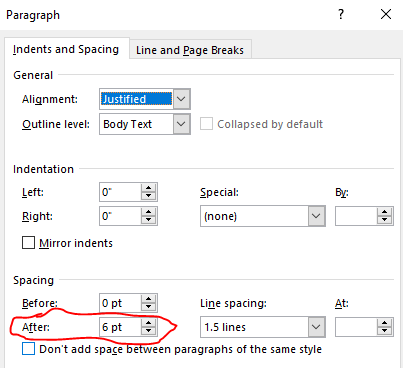


Figure 1.1: Paragraphing style adjustment

Evolutionary techniques like genetic algorithms have significantly impacted various fields because they can find optimal solutions in complex problem spaces. These techniques are widely used in engineering for problems like design optimisation, scheduling, and resource allocation. For instance, genetic algorithms can optimise the design of aerodynamic structures, enhancing performance and efficiency by evolving designs through iterative selection, crossover, and mutation processes ***[End of paragraph 2]***.

Evolutionary techniques have been applied to personalised medicine and drug discovery in healthcare. By simulating the natural evolution process, these algorithms help identify optimal treatment plans tailored to individual patients' genetic profiles, improving the effectiveness of therapies. In drug discovery, genetic algorithms assist in identifying potential compounds by exploring vast chemical spaces, accelerating the development of new medications ***[End of paragraph 3]***.

The financial sector also benefits from evolutionary techniques, particularly algorithmic trading and portfolio optimisation. Genetic algorithms help develop trading strategies that adapt to changing market conditions, maximising returns and minimising risks. They also assist in optimising investment portfolios by selecting the best combination of assets to achieve desired financial goals, considering various constraints and market dynamics ***[End of paragraph 4]***.

## 1.3. Figure and Table Captions

Right-click on the table/figure and select the option to add a caption to the object. A sample for each object (table or figure) is depicted in Figure 1.2 and Table 1.1. Notice that the caption for a figure appears below the object while the caption for a table appears above it.



Figure 1.2: A depiction of the two containers used for our experiment

The caption automatically updates to 1.2 since this chapter has an existing figure (Figure 1.1) when the caption was added. Students only need to select the suitable object (make sure you select "Figure" in the label option instead of "Table") when captioning a figure, and the numbering is automatically set. If the chapter prefix (The "1" in Figure 1.2) was not added, clear the previous caption and repeat the captioning process to apply the setting similar to the depiction in Figure 1.3. In the caption dialogue box, click on numbering and adjust the setting to ensure it looks like the left part of Figure 1.3 for the prefix effect.

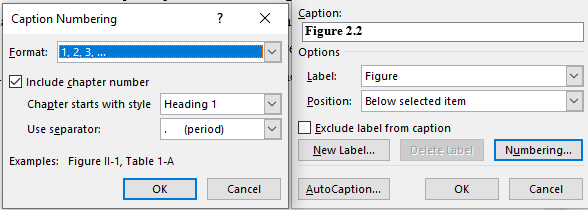


Figure 1.3: Setting the figure caption to carry the chapter prefix (1, 2, ...)

The header should be in the "Table Header" style, while the contents should be in the "Table-Text" style. A sample table is in Table 1.1.

Table 1.1: A non-exhaustive list of Key Algorithms in Computer Science: Pioneering Creations and their Visionary Architects

|  |  |  |
| --- | --- | --- |
| Algorithm | Author(s) | Year Published |
| Dijkstra's Algorithm | Edsger W. Dijkstra | 1959 |
| Quicksort | Tony Hoare | 1960 |
| A\* Search Algorithm | Peter Hart, Nils Nilsson, Bertram Raphael | 1968 |
| Knuth-Morris-Pratt Algorithm | Donald Knuth, Vaughan Pratt, James H. Morris | 1977 |
| RSA Algorithm | Ron Rivest, Adi Shamir, Leonard Adleman | 1978 |
| SHA (Secure Hash Algorithm) | National Security Agency (NSA) | 1993 |
| PageRank Algorithm | Larry Page, Sergey Brin | 1998 |
| Generative Adversarial Network (GAN) | Ian Goodfellow et al. | 2014 |
| ResNet (Residual Neural Network) | Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun | 2015 |
| YOLO (You Only Look Once) | Joseph Redmon, Ali Farhadi | 2016 |
| AlphaGo | DeepMind Technologies | 2016 |
| Transformer | Vaswani et al. | 2017 |
| BERT Algorithm | Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova | 2018 |

## 1.4. The use of Roman numerals for itemisation

Roman numerals are preferred for content that requires itemisation. The ***Item*** Style has been provided for this purpose. The following text shows how the student should (for example) present the Aim (**Text** style) and Objectives (**Item** style) of your study.

The aim of this study is to develop and evaluate deep learning models for predictive maintenance in industrial systems, aiming to improve equipment reliability, reduce downtime, and optimise maintenance schedules. The objectives are as follows:

1. To conduct a comprehensive review of existing literature on predictive maintenance techniques, with a focus on deep learning approaches.
2. To collect and preprocess historical maintenance data, sensor readings, and equipment logs from industrial systems.
3. To extract relevant features from the collected data indicative of equipment health and failure patterns.
4. To develop and implement deep learning models, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), or their variants, for predictive maintenance tasks.
5. To experiment with different architectures, hyperparameters, and optimisation techniques to enhance model accuracy and robustness.
6. To evaluate the performance of the developed models using appropriate metrics such as accuracy, precision, recall, and F1-score.
7. To deploy the trained models in a real-world industrial environment, ensuring compatibility with existing systems and workflows.

## 1.5. Inserting Equations

One of the best ways to insert an equation is to use the Equation Editor tool of Microsoft Word to make the equation look very presentable. The equations should be numbered and must be referenced before they appear, similar to how figures and tables are numbered and referenced. The Equation Editor can be found under the Insert tab of the Microsoft Office environment, or press "**Alt + =**" on your keyboard to initiate an equation. To insert an equation number, follow the following steps:

1. Click inside the equation where the equation number should appear (at the end of the equation, within the equation text bubble).
2. Go to the "References" tab on the Ribbon.
3. Click on the "Insert Caption" button.
4. In the Caption dialogue box, select "Equation" from the "Label" dropdown menu.
5. Replicate what was done in Figure 1.3 for the figure caption, only if the prefix does not appear in the equation number [This step is optional].
6. Check the box that says "Exclude label from caption" to remove the word "Equation" from the caption.
7. Click "OK" to insert the equation number.
8. Place the cursor on the equation and apply the Eqn-Caption style to the equation.
9. Space/adjust the equation and the numbering to make sure the numbering is at the edge of the page and the equation stays at the centre as much as possible.

Examples of sample blocks of text to reference equations and the equations themselves are provided in the following paragraphs.

K-Means is an unsupervised learning algorithm used for clustering. Equation 1.1 represents the objective function, which aims to minimise the variance within each cluster.

Equation 1.2 is a key component of the Transformer model introduced by Vaswani et al. in 2017. It computes the attention scores used to weigh the importance of different words in a sequence for tasks like machine translation and text summarisation. In the equation, represents the **softmax** function.

# 

**LITERATURE REVIEW**

## 2.1 Preamble – A sample Heading 2 and Citations

Since the first paper (Cowling, Kendall & Soubeiga, 2000) that coined the word "hyper-heuristics" to mean creating methods at a higher level than heuristics and metaheuristics to solve a wide range of optimisation problems was published, several advances have been made in the research field till date. **[You can see that there is a citation in the first sentence of this section. In-text citations require only the authors' last names; initials and other names are not permitted]**. Over the years, numerous amounts of research engaging hyper-heuristics have been done, cutting across several problem domains like Timetabling (Ahmed, Özcan & Kheiri, 2015), Bin Packing (Asta, Özcan & Parkes, 2016), Traveling Salesman Problem (Ryser-welch, Miller, Swan & Trefzer, 2016), Boolean Satisfiability; also known as SAT (Lassouaoui, Boughaci & Benhamou, 2019), Permutation Flowshop Scheduling (Song & Lin, 2021), Knapsack Problem (Drake, Özcan & Burke, 2015), Multi-objective optimisation (Almeida, Gonçalves, Venske, Lüders & Delgado, 2020; Cui, Han, Su, Zhang & Guo, 2020; Guizzo, Bazargani, Paixao & Drake, 2017; Yang, Zhang & Li, 2021; Zhang, Ren, Li & Xuan, 2020; Zhou, Yang & Zheng, 2019) and recently to Bioinformatics (Fontoura, Pozo & Santana, 2017). This section extensively shows multiple versions of in-text citations.

## 2.2 Other citation styles

**Narrative citations:** When the student mentions the author's name in the text, s/he should place only the year of publication in parentheses immediately after the author's name. Sometimes, in-text citations require only the year to be placed in brackets, especially in this situation called narrative citations. See some examples in the paragraphs that follow.

Yahyaoui et al. (2015) embedded a VND hyper-heuristic (VND-HH) into an Iterated Local Search (ILS) metaheuristic. The VND-HH constitutes the local search phase of the ILS framework. Each neighbourhood in the VND-HH is ranked, and this rank is updated during the search phase.

Elhag and Özcan (2015) proposed a grouping hyper-heuristic framework to tackle two optimisation problems that can be reduced to a grouping problem: graph colouring and examination timetabling problems. Three sets of heuristics (LLHs) were presented: merge, divide, and change.

A sequence-based heuristic selection strategy (SSHH) was presented in the work of **Kheiri and Keedwell (2017)** and applied to the domain of high school timetabling. The study used move acceptance methods: Only Improving (OI), Simulated Annealing, Great Deluge, Record-to-Record Travel and Late Acceptance (LA). The proposed method is a perturbative selection hyper-heuristic method that was used to improve the quality of the initial solution. **[You would notice in this example that the full "and" is used in the narrative style rather than the use of the symbol "&"]**.

Tian, Li, Zhou, Guo and Liu (2018) addressed a scheduling problem with an ACO hyper-heuristic. They adopted ACO as the high-level strategy based on its effectiveness as a constructive population-based approach. In the ACO-based hyper-heuristic, the term "dynamic decision blocks" was adopted to enable the proposed approach to generate appropriate heuristic rules for the decision blocks.

### **2.2.1 A Heading 3 style**

This style is below Heading 2.2, and it can be seen that the first section number is 2.2.1, signifying the first sub-section of section 2.2.

### 2.2.2 Another Heading 3 style

This is another Heading 3 style.

## 2.3 Automated software engineering

**[This section demonstrates full use of Item, Text, Headings 2, 3, 4 styles]** using logically related content.

Automated software engineering techniques span various methodologies and tools to improve the software development lifecycle. These techniques can be categorised under various sub-sections, each addressing different aspects of software engineering. Here is an overview:

### 2.3.1 Automated Code Generation

One aspect is Model-Driven Development (MDD), which uses models as the primary artefacts of the development process. Examples include UML (Unified Modeling Language) and Domain-Specific Languages (DSLs) - Specialised languages tailored to specific problem domains, such as SQL for database querying or HTML for web content. Another aspect is the use of Template-Based Code Generators.

### 2.3.2 Automated Testing – This has two Heading 4 sections

Automated testing is a method used in software development where automation tools are employed to control the execution of tests and the comparison of actual outcomes with predicted outcomes. This technique allows test scripts to be executed more quickly and frequently, improving accuracy and efficiency in the testing process. It is beneficial for repetitive tasks and regression tests, helping to ensure that changes to the code do not break existing functionality.

#### A. Unit Testing

Unit testing is a software testing method where individual units or software components are tested independently to ensure that each part functions correctly. A unit typically refers to the smallest testable part of an application, like functions, methods, or classes. Developers usually automate and write unit tests as they work on the code to ensure that any changes or additions do not break existing functionality. Some example tools include JUnit for Java, NUnit for .NET, and PyTest for Python.

#### B. Test Automation Frameworks

Test automation frameworks are sets of guidelines, tools, and practices designed to help in the creation and execution of automated test scripts. They aim to simplify the testing process and make it more efficient by providing a structured environment with reusable components. Some popular test automation frameworks include:

1. Selenium: Primarily used for web applications.
2. JUnit: Used for unit testing in Java.
3. TestNG: Similar to JUnit but with more features.
4. Cypress: For end-to-end testing of web applications.
5. Appium: For mobile application testing across platforms.

These frameworks help streamline the writing, maintenance, and execution of test scripts, making automated testing more accessible and manageable.

### 2.3.3 Artificial Intelligence in Software Engineering – This has three Heading 4 sections

Artificial Intelligence (AI) is increasingly integrated into software engineering to enhance various development processes. AI's role in software engineering is growing, offering new tools and methods to improve efficiency, accuracy, and innovation.

#### A. AI-Driven Code Completion

AI-driven code completion tools use advanced machine learning models, often based on deep learning architectures like transformers, to predict and suggest code snippets or completions as developers type. These tools are trained on large code datasets from various sources, learning syntax, common patterns, libraries, and best practices. Some examples include:

1. **GitHub Copilot:** Developed by GitHub in collaboration with OpenAI, Copilot uses the Codex model to suggest entire lines or blocks of code based on the context of the current file.
2. **TabNine:** An AI-powered code completion tool that integrates with various editors and IDEs, providing context-aware suggestions.
3. **Kite:** A tool that offers code completions and documentation lookups powered by AI, supporting multiple programming languages.

#### B. Machine Learning for Bug Prediction

Machine learning for bug prediction involves training models on historical data of code commits, bug reports, and version control history to identify patterns and features indicative of buggy code. These models analyse metrics like code complexity, changes in code, developer activity, and historical bug fixes to predict potential bugs in new code. Examples include:

1. **Bugspots:** A lightweight tool that uses simple algorithms to identify hot spots in a codebase likely to contain bugs based on commit history.
2. **CodeGuru:** An Amazon Web Services (AWS) tool that uses machine learning to identify bugs and recommend fixes in Java applications.
3. **Predictive models in static analysis tools:** Advanced static analysis tools incorporate machine learning models to predict bug-prone areas in the code.

#### C. AI for Software Design

AI for software design leverages machine learning and artificial intelligence techniques to automate or assist in creating software design artefacts. These tools analyse requirements, existing codebases, and design patterns to generate design diagrams, architecture models, or code scaffolding. Some examples include:

1. **Archium:** A tool that assists in software architecture recovery and evolution by analysing source code and creating architecture diagrams.
2. **Design Automation with IBM Rational Rhapsody:** Uses AI to automate parts of the design process, generating UML diagrams and other design artefacts from requirements.
3. **Generative Design Models:** These AI-driven generative models create initial design proposals based on input requirements and constraints, which human designers can refine.

## 2.4. Summary

In this chapter, we presented the various Heading styles and how they are applied. The various forms of in-text citations were also demonstrated, especially the narrative version of in-text citations where only the year of the reference is in parentheses.

# 

**METHODOLOGY OR SYSTEM ANALYSIS AND DESIGN**

## 3.1 Presenting Algorithms

This section demonstrates how an algorithm should be presented in a project report. The presentation format is a table with a single column and multiple rows; one row corresponds to a line of the algorithm. Like tables and figures, an algorithm should be referenced before appearing in the report. The Iteration Limited Threshold Accepting (ILTA) method (presented in Algorithm 3.1) accepts a proposed solution if it is equal to or better than the current solution based on the objective function value. Additionally, it can accept worsening moves if no better solution is generated over several iterations (k).

|  |
| --- |
| : ILTA |
| **Input**: , , |
| **if** |
|  |
|  |
| **else if** |
|  |
| **else** |
|  |
| **if** |
|  |
|  |
| **end if** |
| **end if** |
| **return** |

Algorithm 3.1 is the preferred format for the project report. However, the same algorithm has been presented with the table borders shown in Algorithm 3.2. This version is to provide evidence that the algorithm presentation is just a table with 1 column and many rows. To transform Algorithm 3.2 to Algorithm 3.1, Click /Select the table object and do the following to the table:

1. Remove the left border.
2. Remove the right border.
3. Remove the inside horizontal border.
4. Restore the bottom border of the first row (you have to place the cursor on that row first).

|  |
| --- |
| : ILTA in Unaccepted Table Format |
| **Input**: , , |
| **if** |
|  |
|  |
| **else if** |
|  |
| **else** |
|  |
| **if** |
|  |
|  |
| **end if** |
| **end if** |
| **return** |

## 3.2 More Equations

This section demonstrates more equations, further reinforcing what equation numbers look like based on the chapters in which they appear. Although equations are already present in the template, the first two equations in this chapter are numbered 3.1 and 3.2, similar to the numbering convention for figures and tables.

### 3.2.1 Routing Problem I: Travelling Salesman Problem

Given a set of cities and their distances from one another, the goal is to find a sequence where each city is visited only once, the route ends with the starting city, and the total distance covered is at a minimum. If the distance between two cities and is and captures whether a solution recommends the travelling salesman passes through the path (but not the reverse case), then the objective function of the problem can be expressed as

### 3.2.2 Routing Problem Domain II: Vehicle Routing

The variant of VRP implemented in HyFlex is the vehicle routing problem with time windows (VRP-TW); in this case, a customer must be served within a time frame before the constructed solution is considered valid (Walker, Ochoa, Gendreau & Burke, 2012). All vehicles begin at a central location (the depot), which must be the first and last point visited for a given vehicle route. A unified objective function that encompasses two objectives of VRP-TW (minimising the number of vehicles and minimising the total distance travelled) is defined as follows:

where is the set of vehicles (fleet), is a constant set to 1000 and is the distance covered by a vehicle in its route. The value of is set to a high value to give a higher priority to solutions that use fewer vehicles. Therefore, a solution that uses two vehicles may likely be better (has a lower objective function value) than a solution that uses three vehicles even if more distance is covered by the two vehicles.

## 3.3 Evaluation Schemes

This section is divided into two parts: the first demonstrates another Heading 3 style, while the second demonstrates another table.

### 3.3.1 Testing New Schemes

A time limit to run on each instance on a machine is set for the scheme, and the value (**in secs**) is determined by a benchmark program after sampling the speed of the machine. This procedure creates a level playing field amongst tested schemes, even if tested on different machines.

### 3.3.2 Evaluation Procedures

The first two evaluation procedures (before the last one) discussed in this sub-section have been used in previous studies to evaluate schemes comparatively. The first one (formula one) that is discussed is the most widely used procedure (Adriaensen, Brys & Nowé, 2014; Alanazi, 2016; Choong, Wong & Lim, 2018; Kheiri & Keedwell, 2015; Sabar, Ayob, Kendall & Qu, 2015; Sabar & Kendall, 2015) and it was introduced during the CHeSC 2011 competition.

#### A. Formula One

The Formula One evaluation procedure has its roots in the Formula One racing competition, where competing drivers are awarded points based on their finishing positions after a race. In our context, obtaining the best median objective function value on an instance is like winning a race for a competing method. An illustration of the formula one scoring and ranking on a sample instance of an arbitrary domain is depicted in Table 3.1 **[You can see a table reference here]**. Regardless of the number of competing hyper-heuristics, only the top eight are awarded points on an instance in the following order: **10, 8, 6, 5, 4, 3, 2, 1**; that is, the method that got the best median objective function value (**ofv**) on a given instance gets **10** points, and the eight best gets **1** point.

Table 3.1: Ranking the top ten hyper-heuristics on an arbitrary instance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rank | Hyper-heuristic | Median ofv | Formula 1 score | Points used |
| 1 | A | 1.0 | 9.0 | 10, 8 |
| 2 | B | 1.0 | 9.0 | 10, 8 |
| 3 | C | 3.0 | 5.0 | 6, 5, 4 |
| 4 | D | 3.0 | 5.0 | 6, 5, 4 |
| 5 | E | 3.0 | 5.0 | 6, 5, 4 |
| 6 | F | 4.0 | 2.5 | 3, 2 |
| 7 | G | 4.0 | 2.5 | 3, 2 |
| 8 | H | 5.0 | 0.5 | 1 |
| 9 | I | 5.0 | 0.5 | 1 |
| 10 | J | 6.0 | 0.0 | 0 |

#### B. -norm and -rank

In this section, we define/introduce a new style called Numbered-Item. This style may be used with the Item style (based on Roman numerals). The Numbered-Item should be higher than the Item style, using Hindu-Arabic numerals (1, 2, …) instead of Roman numerals. The two main metrics are discussed alongside two others as alternative evaluation methods besides the Formula One scoring system. These four metrics form parts of a whole evaluation method, i.e. they are typically used together (Almutairi, Özcan, Kheiri & Jackson, 2016).

1. **-norm**: Given a problem domain, let represent a scheme, represent a single instance, represent a set of instances of the given problem domain, and represent the total number of runs made on each domain instance. The normalised ofv of a given run in the range by on an instance , denoted as is computed as follows:

The following variables represent the ofv of a given run by the hyper-heuristic , the ofv of the best run (the least ofv) by any given hyper-heuristic (including **)** and the ofv of the worst run (the highest ofv) by any given hyper-heuristic (including **)** on the given instance respectively. The resulting metric for on the problem domain is computed as follows:

1. **-rank**: Given competing hyper-heuristics on an instance (as defined earlier), the best scheme based on its obtained median ofv gets a rank of **1,** and the worst hyper-heuristic on gets a rank of . The rank of a scheme h is denoted as , therefore the resulting metric for across all the instances is:

#### C. Inserting Figures Revisited

In this section, we demonstrate what adding a figure from another study/source looks like to the report. Firstly, the figure has to be referenced before it appears in the report, as per the usual rule. In the figure caption, the source of the figure has to be referenced, too. We provide two examples of this in the following paragraphs.

In this section, we present what evaluating a scheme with Box-plot visualisations looks like in Figure 3.1.

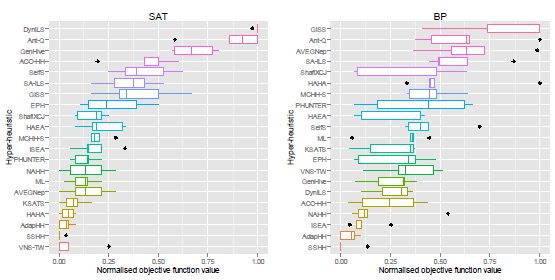


Figure 3.1: A sample box-plot visualisation on two domains [Source: Kheiri & Keedwell (2015)]

The webpage for Internet connectivity login within the Covenant University community is presented in Figure 3.2.

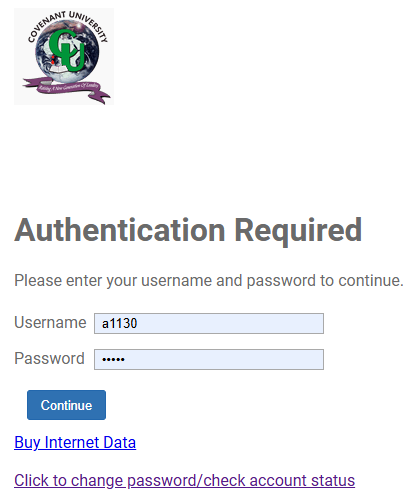


Figure 3.2: Internet Connectivity login page within Covenant University (Source: [internetlogin.cu.edu.ng](https://internetlogin.cu.edu.ng/portal?))

## 3.4 Summary of Styles used in this template

In this section, we describe all the styles relevant to this template to make students aware of them and use them appropriately.

### 3.4.1 Heading 1

This style is used for the chapter numberings – Chapter 1/One, Chapter 2/Two, Chapter 3/Three. The student should not apply it (bar any mistaken altering of the numbering) since the template already contains contents from Chapter One to Chapter Five. However, the student can edit the text behind each Chapter numbering if the title of the chapter is different based on the peculiarity of the work. For example, Chapter 3 might be named "METHODOLOGY" or "SYSTEM ANALYSIS AND DESIGN".

### 3.4.2 Heading 2

This style is used for the main sections of a chapter. Sections 2.1, 2.2, 2.3, 3.4, 4.2, … should use the Heading 2 style.

### 3.4.3 Heading 3

This style is used for the sub-sections of Heading 2 (2.1, 2.2, 2.3, 3.4, 4.2) sections. For example, this section (3.4.3) uses Heading 3.

### 3.4.4 Heading 4

Heading 4 has an alphabetic numbering. It is the third-level section within a chapter of a student's project report.

### 3.4.5 Heading 5

Remember, there was an instruction not to create sections with Heading 5. That instruction is still valid. However, the style can only used for numbering algorithms, and it provides the automatic numbering functionality for algorithms if they appear in the project report. In the table header of the algorithm, type a text and apply the heading style, and the algorithm will be automatically labelled. Note that the contents of the rows of the algorithm presentation should use the Table-Text style since the presentation uses a table.

|  |
| --- |
| : GCD Algorithm |
| **Inputs:** Two integers and |
| **while** |
|  |
|  |
|  |
| **end while** |
| **return** |

### 3.4.6 Numbered-Item & Item

The two styles are for itemisation, but the Numbered-Item style is higher in the hierarchy than the Item style. However, if just one level of numbering is done, use only the Roman numeral style (Item). Let us see an example of how they can be used together:

Rendering algorithms in Computer Science can be categorised based on the rendering they perform. Here are some mentions of the main categories of rendering algorithms:

1. **Rasterisation algorithms**: Rasterisation algorithms are computational methods used in computer graphics to convert vector graphics (defined by geometric shapes such as lines, triangles, and polygons) into raster images (composed of pixels or dots). Some of them include:
2. Scanline Rendering: Processes one scan at a time, filling in pixels between endpoints of each line segment.
3. Z-Buffer Algorithm: Determines visible surfaces by comparing the depths of objects against each other.
4. **Ray Tracing Algorithms**: Ray tracing algorithms are a set of computational methods used in computer graphics to generate highly realistic images by simulating how light interacts with objects in a virtual environment. Unlike rasterisation, which converts geometric shapes into pixels, ray tracing traces the path of light rays as they travel through the scene, allowing for accurate rendering of shadows, reflections, refractions, and other optical effects. Here are some sample algorithms under this category:
5. Whitted-Style Ray Tracing: This classic ray tracing algorithm introduced recursive ray tracing for reflections and refractions. It generates primary rays from the camera, and for each intersection, it generates secondary rays for shadows, reflections, and refractions.
6. Path Tracing: A more advanced form of ray tracing that simulates global illumination by tracing multiple paths per pixel and averaging the results. This algorithm handles complex lighting interactions such as indirect lighting, caustics, and colour bleeding.
7. Metropolis Light Transport (MLT): This advanced ray tracing algorithm uses a Monte Carlo method to explore light paths. It focuses on the most significant paths to improve efficiency and reduce noise.
8. **Global Illumination Algorithm**: Global illumination (GI) algorithms are a class of algorithms in computer graphics that aim to simulate the complex interactions of light in a scene, including direct illumination from light sources and indirect illumination resulting from light bouncing off surfaces. These algorithms produce highly realistic images by accounting for all possible light paths, which include reflections, refractions, and scattering. Here are three prominent global illumination algorithms:
9. Radiosity: Radiosity is a global illumination algorithm that calculates the diffuse interreflection of light between surfaces. It is particularly well-suited for scenes where most surfaces are diffuse (i.e., they scatter light uniformly in all directions).
10. Photon Mapping: Photon mapping is a two-pass algorithm that efficiently simulates global illumination, including caustics, diffuse interreflection, and participating media.
11. Bidirectional Path Tracing: Bidirectional path tracing combines ideas from both path tracing and light tracing to improve convergence rates, especially in scenes with challenging lighting conditions. It traces rays from both the camera and the light sources, connecting them to form light transport paths.

### 3.4.7 Caption

This style is applied when the student adds a caption to a figure.

### 3.4.8 Table-Caption

The style is applied when the student adds a caption to a table.

### 3.4.9 Eqn-Caption

This style is applied when numbering is added to an equation.

### 3.4.10 Text

This particular statement is in text style.

### 3.4.11 Text-Narrow

Apply to the text in the abstract section.

### 3.4.12 Table Header & Table-Text

Table columns use the Table Header style, while the contents of a table use the Table-Text style.

# 

**SYSTEM IMPLEMENTATION AND EVALUATION**

## 4.1 Preamble

This chapter provides instructions on what to do with each section before the first page of Chapter One.

## 4.2 Certification Page

In the prescribed order, fill in the relevant details, including the author's surname, first name, and last name. For the supervisor and the HoD details category, edit the contents if necessary. However, use the prescribed order for the names too: [Title in short form], then [First Name], then [Initial for the Middle Name], and then the [Surname]. The name "Dr. Aderonke A. Oni" fits the name format described earlier.

## 4.3 Dedication Page

There is little to do here except fill in the dedication statement.

## 4.4 Acknowledgements Page

The instructions here are similar to the instructions in section 4.3.

## 4.5 Table of Contents Page

On this page, ensure the cursor is on the area where the actual contents are placed (it should be shaded when the cursor points there). Follow the steps below:

1. Right-click on that area and select "Update Field" to replace the existing table of contents with the contents of your report.
2. Highlight the parts of the table of contents before the line where "CHAPTER ONE" appears (the one with page 12 in Figure 4.1) and apply the **bold font** to these texts.
3. The contents before the main content (Chapter One) are all made bold, and the section is set.

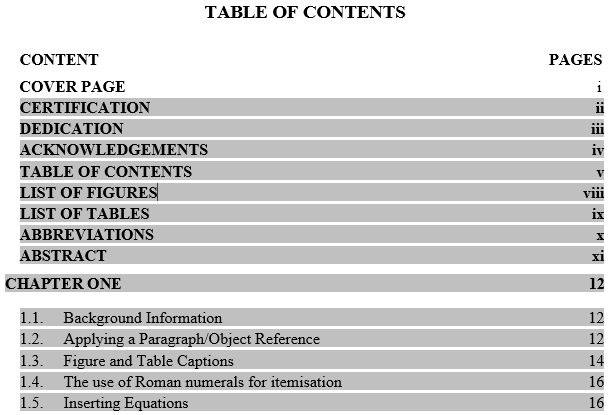


Figure 4.1: Table of Contents when the cursor is on the active area

## 4.6. List of Figures Page

This section is a critical part of using this template, as it does not directly rely on the automatic placement feature provided by Microsoft Word. A demonstration will be done in this section to enable easy understanding of the task. Firstly, it is recommended that a copy of the project report be made to place the Automatic List of Figures in the report directly, or the Automatic List of Figures should be placed at the end of the report (and removed later). The Automatic List of Figures based on the contents of this template is presented as follows, although the figures in the Appendix section have been deliberately left out (this is not right, though):

[Figure 1.1: Paragraphing style adjustment 13](#_Toc169456391)

[Figure 1.2: A depiction of the two containers used for our experiment 14](#_Toc169456392)

[Figure 1.3: Setting the figure caption to carry the chapter prefix (1, 2, ...) 15](#_Toc169456393)

[Figure 3.1: A sample box-plot visualisation on two domains [Source: Kheiri & Keedwell (2015)] 28](#_Toc169456394)

[Figure 3.2: Internet Connectivity login page within Covenant University (Source: internetlogin.cu.edu.ng) 29](#_Toc169456395)

[Figure 4.1: Table of Contents when the cursor is on the active area 34](#_Toc169456396)

The contents of the Automatic List of Figures are then used to fill in the actual List of Figures template, a table of rows and three columns, where represents the number of figures in the report, and the extra row is for the table headers (the column titles).

|  |  |  |
| --- | --- | --- |
| **FIGURES** | **TITLES OF FIGURES** | **PAGES** |
| 1.1 | Paragraphing style adjustment | 13 |
| 1.2 | A depiction of the two containers used for our experiment | 14 |
| 1.3 | Setting the figure caption to carry the chapter prefix (1, 2, ...) | 15 |
| 3.1 | A sample box-plot visualisation on two domains [Source: Kheiri & Keedwell (2015)] | 28 |
| 3.2 | Internet Connectivity login page within Covenant University (Source: internetlogin.cu.edu.ng) | 29 |
| 4.1 | Table of Contents when the cursor is on the active area | 34 |

## 4.7 List of Tables Page

See Section 4.6.

## 4.8 Abbreviations

Replace the contents with the abbreviations of terms in the student's report.

## 4.9 Abstract

Replace the content with the one-page Abstract of the project. The style to be applied is Text-Narrow.

# 

**CONCLUSION AND RECOMMENDATIONS**

This chapter concludes by presenting the expected outline for each chapter.

## 5.1 Chapter One

The contents of Chapter One are as follows:

1. Background Information
2. Statement of the Problem
3. Aim and Objectives of the Study
4. Methodology
5. Significance of the Study
6. Limitations of the Study
7. Project Outline

## 5.2 Chapter Two

The Literature Review chapter is a critical component that provides context and background for the research/project. It demonstrates the student's understanding of the topic's existing research, identifies literature gaps, and positions the study within the broader academic conversation. It contains (not strictly) the following:

1. **Preamble**: It **overviews** and briefly introduces the literature review's main topic and purpose. Also, the **structure of the chapter** can be presented.
2. **Theoretical Framework**: Discuss the **major theories and models** relevant to the research topic. Define key **terms and concepts** that will be used in the study.
3. Review of Key Studies: Identify and discuss the foundational or seminal studies in the field/research topic. Review the most recent studies related to the topic. Highlight the methodologies used in previous research and discuss their relevance to the study. Summarise the main findings and contributions of the key studies.
4. **Themes and Trends**: Highlight any **emerging trends (technologies)** or new research areas in the field. **Discuss any controversies**, debates, or areas of disagreement among researchers.
5. **Gaps in the Literature**: **Point out gaps or** shortcomings in the existing literature/projects/software. Explain how the research/project will **address these gaps** and contribute to the **field/the university community**.
6. **Conclusion**: **Summarise** the key points discussed in the literature review. Reiterate how the **literature review has informed** the research questions or hypotheses or the plan for the project. **Provide a transition** to the project's next chapter, the methodology chapter.

## 5.3 Chapter Three

This chapter provides a detailed account of the research methods or system development processes used in the study. Depending on the most suitable topic for the project (either "Methodology" or "System Analysis and Design", It contains (not strictly) the following:

1. **Preamble**: Briefly introduce the **purpose of the chapter**. State the **objectives of the chapter**, whether it is to describe the research methodology or to detail the system analysis and design process.
2. **Research Design**: Describe the **approach**, whether the study is qualitative, quantitative, or mixed-methods. Specify the **type of study** (e.g., experimental, correlational, case study).
3. **System Analysis**: Clearly **define the problem** the system intends to solve. Identify and describe the needs of the users (**user requirements**). Outline the **functional and non-functional requirements** of the system. **Analyse existing systems** that will be replaced or integrated by the new system. Conduct a **feasibility study** covering technical, economic, legal, operational, and schedule feasibility.
4. **Research Methods**: For the **data collection methods**, describe the methods used for collecting **primary data** (e.g., surveys, interviews, experiments) and explain how **secondary data** was gathered and utilised. Describe the **participants or subjects** involved in the study, including selection criteria and sample size. Detail the **tools and instruments** used for data collection, such as **questionnaires**, **interview guides**, or **software tools**.
5. **System Design**: Describe the system's overall architecture, including high-level design components. Provide **Entity-Relationship diagrams** to represent data models. Describe the **database schema**, including tables, fields, and relationships. Design the **user interface**, including mockups or wireframes. Use **flowcharts** to represent system processes and workflows. Detail any **specific algorithms** used in the system.
6. **Data Analysis Plan**: Describe how the collected data will be **processed and prepared** for analysis. Detail the data analysis methods (e.g., statistical analysis, thematic analysis). Specify any **software or tools** used for data analysis (e.g., SPSS, MATLAB).
7. **Implementation Plan**: Describe the **hardware and software environment** used for system development. Discuss any **coding standards** or practices followed during development. Describe the testing procedure, such as the **unit testing** process and **tools**. Detail how different components were tested together (**integration testing**). Explain how the **complete system** was **tested**.
8. **Ethical Considerations**: Consent, Confidentiality, and Ethical Approval issues.
9. **Conclusion**: **Summarise** the key points covered in the chapter. Provide a **transition** to the next chapter.

## 5.4 Chapter Four

By detailing the development process, testing, deployment, and evaluation, this chapter demonstrates the practical application of the project and assesses its effectiveness, providing valuable insights and recommendations for future work. It contains (not strictly) the following:

1. **Preamble**: Briefly **introduce** the **purpose** of the chapter. State the **main objectives of the chapter**, focusing on the implementation process and evaluation methods.
2. **System Implementation**: The implementation phases begin with the **setup and configuration** of the development environment. Describe the **coding and development phase**. Next is the **integration phase**, where different modules and components are integrated to address any challenges encountered. The **testing phase** should be carried out and described. Finally, the **deployment**, detailing the deployment steps and conducting post-deployment activities such as monitoring and maintenance.
3. **Evaluation**: Evaluate whether the system meets the **functional and non-functional requirements** specified in the design phase. Describe how **user feedback** was collected (e.g., surveys, interviews). Detail the **performance testing** methods and results, including response time and throughput metrics. Explain the **usability testing process and results**, including user satisfaction and ease of use. **Summarise the results** of the evaluation, highlighting key findings. Provide a **detailed analysis of the results**, discussing how well the system meets the specified requirements. Identify the **strengths and weaknesses of the system** based on the evaluation results.
4. **Discussion**: **Interpret the evaluation results**, explaining their significance and implications. **Compare** the developed system with existing systems or solutions. Discuss **any limitations** identified during the implementation and evaluation phases. **Provide recommendations** for improving the system based on the evaluation results and identified limitations.
5. **Conclusion**: **Summarise** the key points discussed in the chapter, including the implementation process and evaluation results. Suggest potential areas for **future work** or further development of the system. Provide a **transition to the next chapter**, which may cover the conclusions and recommendations of the study.

## 5.5 Chapter Five

The contents of Chapter Five are as follows:

1. Summary
2. Conclusion
3. Recommendations

## 5.6 References

The references section contains a list of all the cited works in APA style. The sample references used in this template appear in the list of references on the next page. Apply the **Bibliography** style to the references in the list added to the project report.

## REFERENCES

Adriaensen, S., Brys, T., & Nowé, A. (2014). Fair-share ILS: A simple state of the art iterated local search hyperheuristic. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 1303–1310. ACM.

Adriaensen, S., Ochoa, G., & Nowé, A. (2015). A benchmark set extension and comparative study for the HyFlex framework. *2015 IEEE Congress on Evolutionary Computation, CEC 2015*, 784–791.

Ahmed, L. N., Özcan, E., & Kheiri, A. (2015). Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, *42*(13), 5463–5471. https://doi.org/10.1016/j.eswa.2015.02.059

Alanazi, F. (2016). Adaptive Thompson Sampling for hyper-heuristics. *2016 IEEE Symposium Series on Computational Intelligence*, 1–8. IEEE.

Almeida, C. P., Gonçalves, R. A., Venske, S., Lüders, R., & Delgado, M. (2020). Hyper-heuristics using multi-armed bandit models for multi-objective optimisation. *Applied Soft Computing Journal*, *95*, 106520. https://doi.org/10.1016/j.asoc.2020.106520

Almutairi, A., Özcan, E., Kheiri, A., & Jackson, W. G. (2016). Performance of Selection Hyper-heuristics on the Extended HyFlex Domains. *International Symposium on Computer and Information Sciences*, 154–162. Cham: Springer.

Asta, S., Özcan, E., & Parkes, A. J. (2016). CHAMP: Creating heuristics via many parameters for online bin packing. *Expert Systems with Applications*, *63*, 208–221. https://doi.org/10.1016/j.eswa.2016.07.005

Choong, S. S., Wong, L. P., & Lim, C. P. (2018). Automatic design of hyper-heuristic based on reinforcement learning. *Information Sciences*, *436*, 89–107.

Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. *International Conference on the Practice and Theory of Automated Timetabling*, 176–190. https://doi.org/10.1007/3-540-44629-x\_11

Cui, R., Han, W., Su, X., Zhang, Y., & Guo, F. (2020). A multi-objective hyper heuristic framework for integrated optimisation of carrier-based aircraft flight deck operations scheduling and resource configuration. *Aerospace Science and Technology*, *107*, 106346. https://doi.org/10.1016/j.ast.2020.106346

Drake, J. H., Özcan, E., & Burke, E. K. (2015). Modified Choice Function Heuristic Selection for the Multidimensional Knapsack Problem. *Genetic and Evolutionary Computing*, 225–234. https://doi.org/10.1007/978-3-319-12286-1

Fontoura, V. D., Pozo, A. T. R., & Santana, R. (2017). Automated design of hyper-heuristics components to solve the PSP problem with HP model. *IEEE Congress on Evolutionary Computation (CEC), 2017*, 1848–1855. IEEE.

Guizzo, G., Bazargani, M., Paixao, M., & Drake, J. H. (2017). A hyper-heuristic for multi-objective integration and test ordering in google guava. *International Symposium on Search Based Software Engineering*, 168–174. https://doi.org/10.1007/978-3-319-66299-2\_15

Kheiri, A., & Keedwell, E. (2015). A sequence-based selection hyper-heuristic utilising a hidden markov model. *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, 417–424. ACM.

Lassouaoui, M., Boughaci, D., & Benhamou, B. (2019). A multilevel synergy Thompson sampling hyper-heuristic for solving Max-SAT. *Intelligent Decision Technologies*, *13*(2), 193–210.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., … Burke, E. K. (2012). HyFlex: A benchmark framework for cross-domain heuristic search. *European Conference on Evolutionary Computation in Combinatorial Optimization*, 136–147. Berlin, Heidelberg: Springer.

Ryser-welch, P., Miller, J. F., Swan, J., & Trefzer, M. A. (2016). Iterative Cartesian Genetic Programming: Creating general algorithms for solving Travelling Salesman Problems. *European Conference on Genetic Programming*, 294–310. Cham: Springer.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2015). Automatic design of hyper-heuristic framework with gene expression programming for combinatorial optimisation problems. *IEEE Transactions on Evolutionary Computation*, *19*(3), 309–325.

Sabar, N. R., & Kendall, G. (2015). Population based Monte Carlo tree search hyper-heuristic for combinatorial optimisation problems. *Information Sciences*, *314*, 225–239.

Song, H. B., & Lin, J. (2021). A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times. *Swarm and Evolutionary Computation*, *60*, 100807.

Walker, J. D., Ochoa, G., Gendreau, M., & Burke, E. K. (2012). Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework. *International Conference on Learning and Intelligent Optimization*, 265–276. https://doi.org/10.1007/978-3-642-34413-8\_19

Yang, T., Zhang, S., & Li, C. (2021). A multi-objective hyper-heuristic algorithm based on adaptive epsilon-greedy selection. *Complex & Intelligent Systems*, *7*(2), 765–780. https://doi.org/10.1007/s40747-020-00230-8

Zhang, S., Ren, Z., Li, C., & Xuan, J. (2020). A perturbation adaptive pursuit strategy based hyper-heuristic for multi-objective optimisation problems. *Swarm and Evolutionary Computation*, *54*, 100647. https://doi.org/10.1016/j.swevo.2020.100647

Zhou, Y., Yang, J. J., & Zheng, L. Y. (2019). Multi-Agent Based Hyper-Heuristics for Multi-Objective Flexible Job Shop Scheduling: A Case Study in an Aero-Engine Blade Manufacturing Plant. *IEEE Access*, *7*, 21147–21176.

## APPENDICES [Delete this if not applicable to your report]

Only divide the appendix into sub-sections if there are multiple themes. Use Heading 3 for the sub-sections since Heading 2 is used for the "Appendices" text. However, if there is only one theme, then "Appendices" is changed to "Appendix", and there is no need for multiple sub-sections like Appendix A and Appendix B.

## APPENDIX A: Best Sequences Evolved by Our Scheme

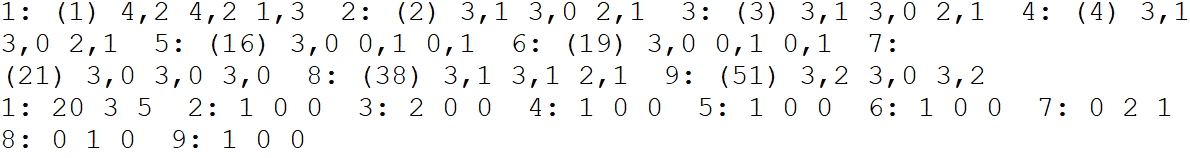


Figure A1: The nine best sequences evolved during one of the best runs of our scheme on SAT3. Objective function value of the best run: 0.0. Dominant sub-schemes: hType-ILS + MC (3, 1) and hType-ILS + APW (3, 0)

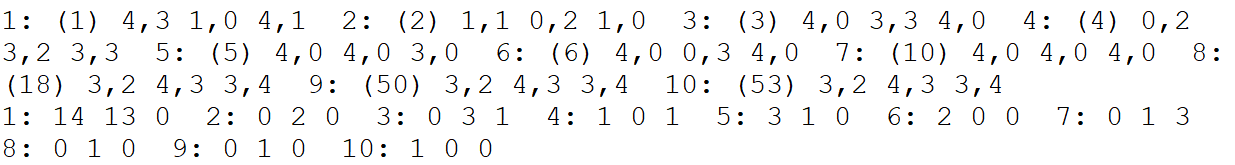


Figure A2: The ten best sequences evolved during one of the best runs of our scheme on SAT5. Objective function value of the best run: 1.0. Dominant sub-schemes: hType-P + APW (4, 0) and hType-P + TA (4, 3)

## APPENDIX B: A C++ Implementation of the sorting algorithm for random keys

void sortKeys(int start) {

fit\_fn\_tsp(start); // compute the fitness values of the

//chromosomes/keys

/\* sort the fitness values of the (biased-)keys using a map insertion

\* mechanism

\* Create a map of the key-value pair

\* where 'key' is a pair of double (the fitness value itself), int

\* (the position it occupies in the array), and 'value' is the

\* original

\* position of the fitness value in the array

\*/

map<pair<double, int>, short> temp;

int iter = 0;

for (short int i = 0; i < n; i++) {

double val = fitness[i];

pair<double, int> p;

p.first = val;

p.second = i;

temp.insert({ p, i });

}

ranks = new int[n];

iter = 0;

temp\_fit\_elite.clear();

for (auto i = temp.begin(); i != temp.end(); i++) {

ranks[iter] = i->second;

if (iter < p\_e)

temp\_fit\_elite.push\_back(fitness[i->second]);

iter++;

}

}