Optimal Task Offloading for Cloud Networked Robotics: A Genetic Algorithm Approach

Akhlaqur Rahman

A thesis submitted in total fulfilment of the requirements for the degree of Doctor of Philosophy (PhD)

2019

Faculty of Science, Engineering and Technology (FSET)
Swinburne University of Technology
Australia
Abstract

The emergence of the Internet of Things (IoT) and cloud computing have produced a paradigm shift leading to the development of integrated robotic applications and services. In order to meet the increasing demand of robot’s energy-intensive applications, the computation-hungry tasks are deployed to the cloud. Thus, task offloading plays a critical role in cloud networked robotics (CNR) for leveraging computation support from the cloud infrastructure. However, considering the delay constraint, the extra costs of data transmission and remote computation, it is not trivial to make optimized offloading decisions. Even though many attempts have been made to study different aspects of offloading, most of them are dedicated towards mobile cloud computing. In reality, offloading process for CNR is more complex due to robot’s on-demand mobility that significantly impacts the relationship among offloading, movement and communication. To address these limitations, it is hence essential to establish more comprehensive offloading techniques during system modelling that are capable of handling higher level of complications. Different from the previous studies that separately emphasised on the aforementioned topics, our approach aims to jointly consider path planning, link selection and offloading as part of the decision-making for different types of CNR systems.

In this thesis, we present a task offloading framework for cloud networked robotics that accommodates mobility and communication as part of its offloading. In order to highlight the impact of the aforementioned parameters on task offloading, we initially present a use case of smart city crowd control application where we design a genetic algorithm (GA) offloading scheme and individually vary robot’s location and bandwidth values in order to assess their impact on the offloading. The outcome not only suggests performance improvement through a cloud-based approach, but also demonstrates the influence of movement and bandwidth on task offloading.
This inspires us to develop a mobility-driven and communication-aware offloading mechanism for a cloud-assisted robot. For our use case of a smart factory maintenance, we formulate a multi-objective optimization where offloading, path planning and AP selection are all considered as decision variables. A GA-based three-layer scheme is then designed in order to solve the problem by identifying the optimal decisions for all three parameters jointly. Comparison with the findings from the reference case (fixed movement and bandwidth) demonstrates a clear progression for our proposed approach due to consideration of mobility (path planning) and communication (AP selection) in its offloading decisions.

For multi-robot cloud networked systems though, there is an added complexity of local (robot-robot) communication, which makes the offloading process more complicated. Since the proposed 3-layer offloading approach for a cloud-assisted robot cannot be directly mapped to the multi-robot network, a separate study is required to accommodate the additional dimension. We tackle this difficulty by partitioning the GA based decision scheme into four-layers (i.e., task offloading, choice of robot for offloading, movement and AP selection) for our joint optimization problem. Simulation is then run for our smart warehouse scenario, where the results suggest that adding local communication as part of a decision set allows the primary robot to distribute tasks locally as well as use other available robots to offload to the cloud. This in turn saves the robotic energy consumption and further improves the offloading process.

Motivated by the limitations of the widely studied offloading process for CNR systems, our proposed GA scheme is the first approach to jointly leverage the unique relationship among the robot’s computational parameters (i.e., offloading, mobility, robot-cloud communication, local communication) and achieve improved performance for different types of CNR systems through its adaptive offloading decision-making.
Acknowledgement

First, I would like to offer thanks to Allah (swt) for giving me the opportunity to do my PhD at Swinburne University of Technology and guiding me with knowledge, strength and endurance throughout my study. I would like to express my sincere gratitude to my supervisor, Dr Jiong Jin for his guidance, encouragement and enduring support for the duration of my research study. I am deeply grateful to him for pushing me to strive for the best and for providing the appropriate research ideas to accomplish my goals.

I greatly appreciate the continuous help from my associate supervisor Dr Antonio Cricenti, who always made time for helping me find opportunities as well as guide me while preparing my thesis, which was really useful. I am also completely indebted to my research mentor and my role model Dr Ashfaqur Rahman for guiding me in every aspect of my research and my life. Specifically, his encouragement in the early stages of my PhD helped me to survive this journey to this present time. This PhD will always be a reminder of his influence and his essential need in my life.

I take this opportunity to sincerely acknowledge the Swinburne University of Technology for providing financial assistance in the form of a PhD scholarship, which buttressed me while I performed my research work. I feel lucky to have been part of an office that comprised really helpful and cheerful people who made this stressful time of my life enjoyable at times. They were a source of great encouragement. Hence, I would to thank Rezwanul, Ovi, Arafat, Aminur, Ragib, Diba, Mahbuba, Nathan and Krish. I would also like to thank my friends at Footscray and Hawthorn, along with people from the rest of Melbourne who provided the support/relaxation that was necessary during parts of this journey.
Finally, I would like to acknowledge my parents, for whom I started this PhD journey. Everything in my life is because of you and this PhD is for you. I must mention my sister, Farzana, who has held my hand in some of the most important stages of this journey to provide encouragement. Last but not least, I am profoundly thankful to my wife, Dr. Anika, for her unconditional love, big sacrifice, care, encouragement and support during the entire journey of this PhD. This journey would not have been possible without your presence beside me. May Allah (swt) give you the best reward in life. My earnest desire is to be able to provide you with the same support when you require it in your life.
Declaration

This research work has been done by the candidate and does not contain any material extracted from elsewhere or from a work published by anybody else. The work for this thesis has not been presented elsewhere by the author for any other degree or diploma.

To the best of the candidate’s knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

All work presented in this thesis is primarily that of the author under the supervision of Dr Jiong Jin. Portions of some chapters have been published in journals and conferences and others are expected to be published also.

Signature:  

Akhlakur Rahman
Melbourne, Australia
September 2018
List of Publications

Papers Published:


Papers (Under Review):


Papers (Under Submission):

# Table of Contents

Abstract ................................................................................................................................. iii
Acknowledgement ................................................................................................................. v
Declaration ............................................................................................................................. vii
List of Publications .............................................................................................................. ix
Table of Contents .................................................................................................................. xi
List of Figures ........................................................................................................................ xv
List of Tables .......................................................................................................................... xix
List of Symbols ........................................................................................................................ xxi
List of Abbreviations .............................................................................................................. xxv

Chapter 1. **Introduction.** ................................................................................................. 1

1.1 Cloud Networked Robotics (CNR) .................................................................................. 2
1.2 Task Offloading ............................................................................................................... 5
1.3 Main Challenges and Motivation ..................................................................................... 8
1.4 Objectives of the Work and Major Contributions .......................................................... 10
1.5 Organization of the Thesis ............................................................................................. 13

Chapter 2. **Task Offloading Optimization for Cloud Networked Robotics** ................. 17

2.1 Optimization Techniques for Task Offloading ............................................................. 17
2.2 Task Offloading in Cloud Networked Robotics ............................................................. 25
2.3 Implications of Communication and Mobility ............................................................... 32
   2.3.1 *Communication aspects* ..................................................................................... 32
   2.3.2 *Mobility aspects* ................................................................................................. 36
2.4 Reasoning for the Genetic Algorithm Approach .......................................................... 38
2.5 Positioning of the Contributions ................................................................................... 40
   2.5.1 *An integrated task offloading framework for CNR applications* ....................... 42
   2.5.2 *Mobility-driven and communication-aware offloading for cloud-aided robot* ........ 42
   2.5.3 *Energy-efficient task offloading for multi-robot cloud networked system* ....... 44
4.6.2 Offloading performance comparison ............................................................... 123
4.6.3 Energy-distance weighted sum for fitness score .............................................. 125
4.6.4 Recharge-based offloading and path planning ................................................. 127
4.7 Summary ............................................................................................................. 128

Chapter 5. Energy-Efficient Optimal Task Offloading for Cloud Networked Multi-
Robot Systems ........................................................................................................ 131

5.1 Introduction .......................................................................................................... 131
5.2 Task Offloading Mechanism for Multi-Robot Systems ....................................... 135
5.3 Application Use Case: Smart Warehouse Parcel Management .............................. 137
5.4 Joint Optimization Problem Formulation ............................................................. 144
  5.4.1 System modelling ............................................................................................. 144
  5.4.2 Cost functions ................................................................................................ 151
5.5 GA-based 4-Layer Decision-Making Scheme ...................................................... 156
5.6 Simulation Results and Analysis ......................................................................... 166
  5.6.1 Analysis of decision-making scheme .............................................................. 170
  5.6.2 Comparative performance evaluation ............................................................ 177
5.7 Summary ............................................................................................................. 180

Chapter 6. Conclusion and Future Work .................................................................. 183

6.1 Summary of Contributions .................................................................................. 183
6.2 Recommendations for Future Work .................................................................... 186
  6.2.1 Real-time resource allocation to deal with dynamic network structure ............ 186
  6.2.2 Task allocation for cloud robotic services by introducing edge
      resources .............................................................................................................. 188
  6.2.3 Virtual machine (VM) migration in cloud networked robotics ...................... 189

References ................................................................................................................ 191
List of Figures

Figure 1.1: Different applications of cloud networked robotics (CNR) [13] .................. 3
Figure 1.2: Features of task offloading in cloud robotics (Reproduced from [16]) .......... 6
Figure 2.1: A sample task graph (DAG) for task offloading optimization [34] .......... 20
Figure 2.2: Overall process design of a genetic algorithm-based (GA) approach
   (Adopted from [65]) ........................................................................................................ 23
Figure 2.3: Dynamic collaboration between underwater aerial vehicles and cloud from
   Pandey et al. [105] ........................................................................................................ 27
Figure 2.4: Data retrieval framework of cloud robotic system from Wang et al. in
   [109] .............................................................................................................................. 28
Figure 2.5: An energy sensitive GA-based task offloading mechanism proposed by Guo
   et al. in [116] .................................................................................................................. 29
Figure 2.6: List of differences between cellular network and WiFi for cloud networked
   robotics (reproduced from [29]) .................................................................................. 31
Figure 2.7: Relationship among task offloading, path planning and AP selection .......... 41
Figure 2.8: Simple structure of a multi-robot cloud networked robotic system .......... 44
Figure 3.1: Different cloud robotics model (Reproduced from Hu et al. [11]) .......... 48
Figure 3.2: Components of a task offloading framework for cloud networked robotics 50
Figure 3.3: A smart city cloud robotics application of crowd control [182] .......... 56
Figure 3.4: A 20-node operation taskflow of smart city crowd control [183] .......... 57
Figure 3.5: A flow-chart of the genetic algorithm-based (GA) scheme ....................... 66
Figure 3.6: Encoding chromosome (single-layer) for task offloading decision-making 67
Figure 3.7: Pseudo-code for calculating the total robotic energy ............................... 68
Figure 3.8: Pseudo-code for calculating the total task completion time/delay .......... 69
Figure 3.9: Crossover phase for GA-based offloading scheme .................................. 71
Figure 3.10: Mutation phase for GA-based offloading scheme ................................. 72
Figure 3.11: Performance graph of 20-node taskflow (Minimise: $E_{total}$) .......... 75
Figure 3.12: Performance graph of 20-node taskflow (Minimise: $T_{total}$) .......... 79
Figure 3.13: Impact of bandwidth change on minimum completion time (min: $T_{total}$) 82
Figure 3.14: Impact of bandwidth change on offloading decisions (min: $T_{total}$) .......... 83
Figure 3.15: Optimal task sequence for a multi-taskflow offloading problem ................. 86
Figure 4.1: Mobility-driven and communication-aware task offloading mechanism for a  
cloud-assisted robotic system ................................................................................. 97
Figure 4.2: An integrated framework for automated smart factory maintenance (Using  
the case of an oil factory in a remote location) [189] ............................................. 99
Figure 4.3: A 30-node application taskflow for smart factory maintenance ............... 101
Figure 4.4: A 36-cell grid derived from the proposed oil factory environment ........... 104
Figure 4.5: Chromosome encoding for a GA-based 3-layer decision-making scheme 114
Figure 4.6: Pseudocode for robotic energy, time and distance calculation .............. 115
Figure 4.7: Crossover phase for GA-based offloading scheme ................................. 116
Figure 4.8: Mutation phase for GA-based offloading scheme ................................ 118
Figure 4.9: GA-based path planning results under normal scenario ........................ 122
Figure 4.10: Fitness performance for GA scheme (Min: $E_{total}$) .............................. 122
Figure 4.11: Energy performance of energy-distance weighted sum for fitness score  
(with respect to initial battery life) ........................................................................ 124
Figure 4.12: Distance and time performance of energy-distance weighted sum for fitness  
score (with respect to initial battery life) ............................................................... 125
Figure 4.13: Visual representation of a 36-cell workspace for an oil industry with  
recharge-based path planning results (parts a, b and c) ...................................... 127
Figure 5.1: Task offloading mechanism for cloud networked multi-robot system with  
collective consideration for mobility, communication and local sharing .. 135
Figure 5.2: Components of an automated warehouse environment .......................... 137
Figure 5.3: Details of an automated parcel sorting and distribution application in a smart  
warehouse ............................................................................................................. 139
Figure 5.4: Visual representation of taskflow, location constraints and workspace.... 141
Figure 5.5: Steps of GA-based multi-layer decision-making scheme ...................... 157
Figure 5.6: Modified chromosome encoding for GA-based 4-layer decision-making  
scheme for cloud networked multi-robot system ................................................. 157
Figure 5.7: Pseudo-code for robotic energy, time and distance calculation .......... 161
Figure 5.8: Crossover phase for GA-based offloading scheme ............................... 164
Figure 5.9: Mutation phase for GA-based offloading scheme ............................... 165
Figure 5.10: Fitness performance and offloading decisions of GAMRC scheme........ 171
Figure 5.11: Path planning performance of each robot for GAMRC scheme.......... 174
Figure 5.12: Visual representation of AP selection for the offloaded tasks.......... 175
Figure 5.13: Path plan performance comparison of robot $R_1$ using all three methods. 176
Figure 6.1: Traits of computing models for cloud robotics (Hu et al., [11]).......... 186
Figure 6.2: Architecture of cloud-fog computing system (Pham & Huh [222])....... 187
Figure 6.3: A model of cloudlet-based mobile cloud computing system (collected from
Liu et al. [223])................................................................. 188
Figure 6.4: VM migration patterns in a MCC system (Gkatzikis & Koutsopoulos
[225]) .............................................................................. 189
List of Tables

Table 3.1: Notation ........................................................................................................................................... 60
Table 3.2: Simulation parameter setup for the 20-node taskflow ................................................................. 74
Table 3.3: Performance comparison for 20-node taskflow (Minimise: $E_{total}$) ........................................... 75
Table 3.4: Performance comparison for 20-node taskflow (Minimise: $T_{total}$) ........................................... 79
Table 3.5: Impact of bandwidth change on task offloading ............................................................................. 82
Table 3.6: Impact of movement on offloading decisions and system performance ........................................ 85
Table 3.7: Performance of GA-based offloading for each taskflow (min: $E_{total}$) ....................................... 89
Table 3.8: Simulation results for multi-taskflow optimal task sequence ....................................................... 89
Table 4.1: Related work for offloading, AP selection and path planning ....................................................... 94
Table 4.2: Additional notation (following up from Table 3.1) ..................................................................... 109
Table 4.3: Parameter setup for simulation .................................................................................................... 120
Table 4.4: Performance of GA-based decision-making scheme ................................................................. 121
Table 4.5: Performance comparison for offloading (Min: $E_{total}$) .............................................................. 123
Table 4.6: Robot performance for recharge-based offloading ....................................................................... 127
Table 5.1: Additional notations (following on from Table 3.1 and Table 4.2) ............................................. 150
Table 5.2: Simulation parameter setup ......................................................................................................... 168
Table 5.3: Analysis of GA-based decision-making scheme ........................................................................... 169
Table 5.4: Performance of each robot for GAMRC approach ...................................................................... 172
Table 5.5: Task offloading performance comparison (Min: $E_{total}$) ............................................................ 177
Table 5.6: Comparison of robots’ performance among the three methods (Min: $E_{total}$) ......................... 179
### List of Symbols

- \( E_{total} \): Total Task robotic energy (J)
- \( T_{total} \): Total Task completion Time (sec)
- \( E_{Limit} \): Robotic energy Constraint (J)
- \( T_{deadline} \): Delay/Time Constraint (J)
- \( E_R \): Total robotic energy for a task taking place on robot (J)
- \( E_C \): Total robotic energy for a task taking place on cloud (J)
- \( E_{Mov} \): Movement task energy for robot (J)
- \( E_{RC} \): Computation energy for task on robot (J)
- \( E_{idle} \): Computation energy for task on cloud (J)
- \( E_U \): Data upload energy for robot (J)
- \( E_I \): Robot energy for sending instruction to cloud (J)
- \( E_{WSN} \): Robot energy for communication with WSN (J)
- \( v(t) \): A task set for WSN communication
- \( P_i \): Robot processing power during sending instructions to cloud (W)
- \( P_u \): Robot processing power for uploading data to cloud (W)
- \( BPI \): Bits per instruction
- \( N(t_i) \): Number of instructions for a task
- \( \beta \): Bandwidth (kbps)
- \( P_r \): Robot processing power for computation on robot (W)
\( P_{idle} \)  
Robot processing power for computation on cloud (W)

\( d(t_i) \)  
Uploaded data for a task on cloud

\( CPI \)  
Clock cycles per instruction

\( S_r \)  
Clock speed of robot processor

\( S_c \)  
Clock speed of Cloud VM processor

\( P_d \)  
Robot processing power during communication with WSN (W)

\( d_d \)  
Transferred Data during communication with WSN

\( T_r \)  
Data transfer rate during communication with WSN

\( m(t) \)  
A task set for robot movement

\( P_{mov} \)  
Robot processing power during movement (W)

\( R_v \)  
Robot movement velocity

\( T_R \)  
Total Time for a task taking place on Robot (sec)

\( T_C \)  
Total Time for a task taking place on cloud (sec)

\( T_{Mov} \)  
Total time for movement task (sec)

\( T_{WSN} \)  
Total time for communication with WSN (sec)

\( T_{RC} \)  
Total computation time for a task on robot (sec)

\( T_U \)  
Total time for uploading data to cloud (sec)

\( T_{CC} \)  
Total computation time for a task on cloud (sec)

\( T_I \)  
Total time for sending instructions to cloud (sec)

\( I_{t_i} \)  
Task \( t_i \) is executed on robot.
\( \neg l_{t_i} \)  Task \( t_i \) is executed on cloud VM

I  A proposed chromosome/solution

\( f \)  Fitness score of a chromosome/solution

\( \alpha \)  Access point

\( D_{total} \)  Total distance covered by robot

\( \beta(l, \alpha) \)  “Fair-share” bandwidth of robot at location \( l \) with access point \( \alpha \)

\( U_\alpha(t) \)  Set of users connected to AP

\( c_u \)  Cumulative bit rate for the set of users

\( \mathcal{L}_{t_i} \)  Location for each task where the set consists of total \( l \) possible values

\( \mathcal{A}_{t_i} \)  Selected AP for each offloaded task, where AP set has of total \( \alpha \) values

\( \omega \)  Weighing parameter for fitness

\( \mathcal{R} \)  A group of robots for multi-robot system

\( R_i \)  Selected robot

\( l_0 \)  Threshold distance

\( l' \)  Distance between two robots

\( E_{LO} \)  Local communication energy between robots

\( T_{LO} \)  Local communication time/latency between robots

\( e_{base} \)  Baseline energy consumption for operating the transmitter radio

\( \varepsilon_{fs,l'^2} \)  Transmission energy consumption for \( l' < l_0 \)

\( P_{LO} \)  Processing power of for robot \( (R_i) \) local offloading
\( \epsilon_{mf} \cdot l'^4 \)  
Transmission energy consumption for \( l' \geq l_0 \)

\( D_{total}(R_r) \)  
Distance covered for robot \( R_r \)

\( E_{R_r} \)  
Energy consumption of robot \( R_r \)

\( E_{limit}(R_r) \)  
Energy limit of robot \( R_r \)

\( D_{total}(R_r) \)  
Total distance covered by the robot \( R_r \)

\( D_{total}(R_r) \)  
Total distance covered by the robot \( R_r \)

\( P_i(R_r) \)  
Robot \( R_r \) processing power for sending instruction to cloud

\( P_u(R_r) \)  
Robot \( R_r \) processing power for uploading data to cloud

\( P_r(R_r) \)  
Robot \( R_r \) processing power for on-board computation

\( P_{cc}(R_r) \)  
Robot \( R_r \) processing power during cloud computation

\( P_{mov}(R_r) \)  
Robot \( R_r \) processing power during robot movement

\( P_d(R_r) \)  
Robot \( R_r \) processing power for WSN communication

\( v(R_r) \)  
Robot \( R_r \) movement velocity

\( T_r(R_r) \)  
Robot \( R_r \) transfer rate for WSN communication

\( S_{R_r} \)  
Clock speed of robot \( R_r \) processor

\( BPI(R_r) \)  
Bits per instruction for robot \( R_r \)

\( CPI(R_r) \)  
Average number of clock cycles per instruction for robot \( R_r \)
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IoT</strong></td>
<td>Internet of Things</td>
</tr>
<tr>
<td><strong>CNR</strong></td>
<td>Cloud Networked Robotics</td>
</tr>
<tr>
<td><strong>QoS</strong></td>
<td>Quality of Service</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td><strong>ES</strong></td>
<td>Exhaustive Search</td>
</tr>
<tr>
<td><strong>DAG</strong></td>
<td>Direct Acyclic Graph</td>
</tr>
<tr>
<td><strong>AoR</strong></td>
<td>All on Robot</td>
</tr>
<tr>
<td><strong>MCC</strong></td>
<td>Mobile Cloud Computing</td>
</tr>
<tr>
<td><strong>AWS</strong></td>
<td>Amazon Web Service</td>
</tr>
<tr>
<td><strong>WSN</strong></td>
<td>Wireless Sensor Network</td>
</tr>
<tr>
<td><strong>VM</strong></td>
<td>Virtual Machine</td>
</tr>
<tr>
<td><strong>SaaS</strong></td>
<td>Software as a Service</td>
</tr>
<tr>
<td><strong>IaaS</strong></td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td><strong>PaaS</strong></td>
<td>Platform as a Service</td>
</tr>
<tr>
<td><strong>ICT</strong></td>
<td>Information and Communication Technologies</td>
</tr>
<tr>
<td><strong>COP</strong></td>
<td>Common Operating Picture</td>
</tr>
<tr>
<td><strong>UAV</strong></td>
<td>Unmanned Aerial Vehicles</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td><strong>ES</strong></td>
<td>Exhaustive Search</td>
</tr>
<tr>
<td><strong>Acronym</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td><strong>AoR</strong></td>
<td>All on Robot</td>
</tr>
<tr>
<td><strong>HSE</strong></td>
<td>Health, Safety and Environment</td>
</tr>
<tr>
<td><strong>IMR</strong></td>
<td>Inspection, Maintenance and Repair</td>
</tr>
<tr>
<td><strong>GAVM</strong></td>
<td>GA Scheme with Variable Movement</td>
</tr>
<tr>
<td><strong>GAFM</strong></td>
<td>GA Scheme with Fixed Movement</td>
</tr>
<tr>
<td><strong>AoFR</strong></td>
<td>All on Fixed Resources</td>
</tr>
<tr>
<td><strong>CNMRS</strong></td>
<td>Cloud Networked Multi-Robot System</td>
</tr>
<tr>
<td><strong>MRTA</strong></td>
<td>Multi-Robot Task Allocation</td>
</tr>
<tr>
<td><strong>AI</strong></td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td><strong>GAMRC</strong></td>
<td>GA scheme for multi-robot with cloud</td>
</tr>
<tr>
<td><strong>GASRC</strong></td>
<td>GA scheme for a single cloud-aided robot</td>
</tr>
<tr>
<td><strong>GAMRB</strong></td>
<td>GA scheme for multi-robot on-board</td>
</tr>
</tbody>
</table>
Chapter 1.

Introduction

In recent years, the emergence and rapid development of cloud technology and the Internet of Things (IoT) have elevated the potential of integrating autonomous robotic sensing and actuation in dynamic as well as complex applications. To compensate for the highly challenging and customized application demands, much research over the last few years has focused on developing collaborative robots for service-based applications in modern society. This has led to the idea of “networked robotics”, which is defined as a group of robotic devices connected via a wired or wireless communication network to accomplish a common goal. They are also classified as tele-operated robots or multi-robot systems. Any device that requires support from a network like this is considered as part of networked robots. The reason to define this in such a way is to include all the future systems and the existing systems such as UAVs [1] [2] or warehouse robots [3] as well as assembly lines, home automation systems and some specific systems where computation is performed by humans [4]. Such systems have a multitude of applications e.g., industrial support, control of planetary rovers, medical surgery, service-based operations etc. In networked robotics, the workload of sensing, actuating, communicating and computing is distributed among a group of robots. Hence, their deployment is now possible in tedious and potentially dangerous tasks because of the attributes of high endurance, speed and precision. Beginning with analytical tasks such as scene analysis [5], 3D path planning for search and rescue [6], navigation tasks [7] to other interactive tasks, e.g., scene recognition [8], 3D printing, medical surgery [9], all of these are examples of services being provided by the highly equipped modern robot. Even though
the progress made thus far has been exemplary, still there are limitations. Despite all the advancement of late, it is still infeasible to prepare individual robots with limitless capabilities. Every robotic system is bounded by constraints, whether hardware or software. These include system resource constraints, communication constraints and learning constraints etc.

The hardware improvements of robots are limited to an extent by Moore's law. This is because, increasing the clock speed and battery capacity for robots may result in an increase (octuplet) in power consumption too [10]. Therefore, hardware-based approach is not an ideal solution as there are not a whole lot of upgrades we can successfully make in that regard without significant compensation. Despite this challenge, there is still much scope for development as far as software advances are concerned. One such approach is to “offload” the computation/task altogether to a remote resource in order to save energy, shorten response time and extend the battery life of the robot. Having originated from the concept of mobile cloud computing, task offloading has also transitioned to robotic networks as well, thus introducing “cloud networked robotics”. It presents the perfect blend of a robotics network with additional support from cloud infrastructure that has the potential to improve the performance for various service-based applications.

In the following section, we introduce the topic of CNR with task offloading, and then explain our major contributions to this domain.

1.1 Cloud Networked Robotics (CNR)

The paradigm, known as “cloud networked robotics”, refers to an evolutionary upgrade from networked robotics designed to overcome its limitations by leveraging the benefits of cloud computing technologies [11]. First coined by J. Kuffner in 2010 [12], CNR has become a prominent area that has merged the two ever-progressing concepts of
Firstly, the added feature of cloud infrastructure implies less dependence on human input and more support from the ubiquitous virtual resources of the cloud. On the other hand, networked robotics have different constraints such as resource, skill-learning and communication. The ubiquitous and on-demand services provided by “the cloud” [14] enable the robots to access storage and computing resources in order to overcome their own limitations and perform more dynamic and heavy-computation tasks. To access these cloud services, the users must enter into a contract with the cloud service provider, which is known as the Service Level Agreement (SLA). By using these cloud services, the robot offloads tasks through virtualization and use the on-demand computation support (provided by cloud virtual machines) to lessen the workload of the on-board machine.

Another concept imperative for the advancement of CNR applications is the Internet of Things (IoT). This term refers to the ability of everyday objects (equipped with
ubiquitous computing) to communicate with each other, resulting in a highly distributed network of devices. In fact, the emergence of CNR and IoT has elevated the potential of integrating autonomous sensing and actuation in the evolving dynamic and complex applications (as seen in Fig 1.1). Due to all the attributes of virtualization, decentralization and real-time capability, CNR is envisioned to be a key enabler for the infusion of robotic technologies, especially in automating applications such as sensing, actuating and monitoring via the addition of cloud computing and wireless sensors. In fact, CNR encapsulates the design principle of robotic resources integrated with the cheaper computing cost and network resources that have extended their operational capabilities. In doing so, CNR has also produced a shift in the modes of the robot-based applications, [15] from carrying out many repetitive service-based tasks to solving more complex multi-objective problems in uncertain and dynamic environments.

There are four major benefits of cloud computing in networked robotics [13]. They are: a) access to a big library of data, b) computation support, c) collective robot learning and d) human computation support to the user via the cloud. These features have led to several resource management/ allocation studies for cloud computing in this context. The main criterion for the classification of such studies is based on the dynamics of the problem. For example, in the case of static/offline problems, the full list of requests is known a priori. Contrary to that, resource demands become apparent over time for online/dynamic problem sets. Regardless of the problem type, the key enabling factor considered for such an allocation-based approach is the robot’s ability to access the benefits of computation support from the cloud infrastructure, which is possible by “offloading” the task to the cloud. Along with the maturity of cloud networked robotics, task offloading has become a widely used technique for increasing the limited capacity of a robot (energy, battery life, time) by sending computationally expensive tasks to the
redundant, inexpensive and scalable cloud servers. Therefore, more insight is required on

task offloading and the factors (e.g., cost, trade-off etc.) related to its decision-making.

1.2 Task Offloading

As more and more devices become increasingly capable of connecting to the cloud, resource-constrained devices potentially offloads tasks to speed up task execution as well as to utilize computation support. NIST [14] defines cloud computing as a model that enables ubiquitous and on-demand network access to a shared pool of configurable computing resources (such as networks, servers, storage etc.). In doing so, these facilities can then be rapidly provisioned and released with minimal service provider interaction or management effort, thus providing different ranges of services to the users. As previously mentioned, one of the major enabling factors of cloud computing is task offloading. It allows embedded devices with low energy or processing power to access the supporting features of cloud infrastructure and perform computation-intensive tasks with much better efficiency. Thus, task offloading to the cloud infrastructure is a potential approach toward saving local resources (i.e., device energy, task completion time, battery life etc.) as well as improving the system performance.

For robotic applications, cloud computing provides the computation and storage assistance for resource-hungry tasks in order to improve the system outcome. The key objective in such operations is to minimize energy/delay while maintaining the Quality of Service (QoS). Some scientific studies are currently being performed to find energy-saving models and extend the battery life of applications by reducing energy consumption. These are divided into four categories as follows [10]:

- Developing robots with innovative technology of smaller sized semiconductors for low energy purpose. However, additional functionalities and optimal performance for
robots means the need for more transistors. Unfortunately, this proportionately increases energy consumption.

- Programming individual robot components or the complete robot to be in sleep mode/standby mode during the period it remains idle. This prevents the robot from wasting any energy.
- Introducing energy-optimal execution policies that consider the optimal clock frequency of a processor as a key parameter in order to complete given tasks within a predetermined time period. This allows the robot to reduce energy consumption by slowing down the clock frequency and also increases the execution time of particular tasks.
- Transferring the computation burden completely from the robotics. All the necessary and heavy computation are offloaded into the cloud server. Thus, the energy consumption of the robot is mainly for sending and receiving the required information to the cloud server as well as during the period when the robot is idle. This approach for saving energy is also called computation/task offloading. Our study revolves around this topic in the context of CNR.

![Figure 1.2: Features of task offloading in cloud robotics (Reproduced from [16])](image)
A visual representation in Fig. 1.2 highlights the key features of task offloading in cloud robotics applications. As seen in the figure, task offloading in cloud robotics allows the robot to use the storage and computation support to lower the hardware cost of the system as well as to save energy. However, it is not imperative to offload every task. In fact, not every offloading task can be beneficial. For example, offloading to the cloud evokes communication energy between the robot and the cloud. There are also additional factors such as network connectivity, robot movement, robotic capacity etc. All these issues make the process of offloading decision-making complicated for the CNR operations. Hence, a trade-off is required while considering the task offloading decisions. A simple mathematical explanation is presented below to explain the concept of task offloading:

The energy required for the robot to complete a single task \( t \) is \( E_{\text{robot}} \). It takes the robot \( T_{\text{robot}} \) seconds to complete the task on-board. However, if offloaded, the same task takes \( T_{\text{cloud}} \) seconds to be completed on the faster processors of the cloud virtual machine (VM). During the time \( T_{\text{cloud}} \) (idle period for robot), the amount of energy usage for the robot is \( E_{\text{idle}} \). The communication energy required to transfer the instructions and data for the task (to and from the cloud) is considered as \( E_{\text{transfer}} \). So, based on that representation,

If \( E_{\text{robot}} > E_{\text{idle}} + E_{\text{transfer}} \), then the task offloading to cloud would save energy.

If \( E_{\text{robot}} < E_{\text{idle}} + E_{\text{transfer}} \), then the task completed on robot would save energy.

In this way, offloading will not only ensure that the task offloading saves energy, but it will also reduce the task completion time as well. Therefore, this is the basic concept of task offloading for a single task. However, for a set of tasks (taskflow), there are possible combinations, priorities and seriality that needs to be maintained. Moreover, in
the case of multiple parallel tasks, offloading presents an opportunity to better utilize the resources. With all these issues considered, it is imperative to identify the key tasks to offload in a large-scale application. For CNR though, the special features of a robot’s on-demand mobility mean offloading decision process benefits from the added attributes of the robotic system, which are: choice of movement and selection of communication links. Considering these factors, a specific and more concrete study is required to focus on the unique key parameters that influence the task offloading decision-making in different types of cloud robotic applications.

1.3 Main Challenges and Motivation

While task offloading in cloud robotics has been widely considered for the purpose of saving robot energy and execution delay/time, there are still many challenges, which may significantly impede future improvement in system QoS. The main factors worth considering while making task offloading decisions in a CNR application are:

i) Applicability: Although every application consists of multiple tasks, not all tasks are worth offloading. Some tasks may be more suitable for offloading than others (based on their performance trade-off). At the same time, there are some tasks that are worth completing locally, as cloud communication may be too time- and energy-consuming. Depending on the orientation of the task graph, many of these decisions may vary. Since offloading every task in the application is not feasible, tasks must be identified to be offloaded based on the criteria of the task graph as well as the requirement of the application. These factors have serious impact on the offloading decision-making.

ii) Availability and utilization of resources: Another important factor for offloading is the availability of resources. In the context of this study, the resources considered for task
allocation are the robots and cloud virtual machines. During the course of the application, there may be instances when the resources are not available. For multi-robot applications, such issues can be avoided by offloading the task to another robot, so that it is completed locally or offloaded to the cloud. However, for single robot applications, the workload may need be shared only between the robot and cloud VM. Furthermore, for parallel tasks, offloading may be helpful in order to better utilize the resources available at a given time. In all these situations, offloading decisions may be dependent on the availability/selection of resources, which ultimately results in the best outcome for the application (lower energy consumption, faster execution, improved QoS etc.).

iii) Accessibility of cloud/communication link: As previously mentioned, offloading decisions require a trade-off of robotic computation and cloud computation with the communication between the two entities. Thus, the communication links play a major role in offloading decision-making. Since the robots select the access point (AP) to communicate with the cloud, so the availability of APs makes the cloud infrastructure accessible to the user (robot). For large-scale operations, unavailability of APs may cause tasks to be completed locally. Additionally, depending on the choice of access point and the positioning of the robot, communication links between cloud and robot may also vary. Therefore, accessibility of the appropriate communication link (through selection of the AP) is an important criterion for making the offloading decisions.

iv) Coordination among multi-agents: For multi-agent applications, the communication between local resources may help with the offloading of tasks to the cloud, especially when parallel tasks are being completed and available resources (robots) are required to offload tasks through suitable APs to the cloud VM. In such cases, coordinating the robots is an important way of making sure communication between the local resources does not evoke additional energy/time. For example, issues such as distance between robots or
unavailability of robots may cause local intermediate communication to fail, which eventually hampers the offloading of tasks to the cloud.

Consequently, to enable task offloading, the aforementioned issues need to be addressed. While many offloading approaches currently exist for both single and multi-robot systems in the current state of the art (presented later in chapter 2), there are still many challenges that need to be addressed when applying these mechanisms to the robotic applications. As most of the proposed methods are suitable only for mobile cloud computing, they are either application-specific or limited by the hardware constraints of a mobile device. In comparison, robotic systems operate with a higher power level and they have key criteria that separate them from mobile cloud computing (MCC), which is why most MCC offloading algorithms cannot be directly mapped to a CNR application. More specifically, the key distinction between MCC and CNR is the robot’s unique ability to move on-demand (unlike mobiles), which allows them to actively access better communication links for offloading through the usage of its movement. Even for multi-robot systems, the movement of robots determines their positioning along with their coordination with respect to other robots, which eventually helps with local offloading as well as cloud-based offloading. In summary, it is important to investigate the role that both mobility and communication can actively play in decision-making of task offloading for a CNR application. Hence, a detailed study is required to focus on their impact on the application as well as identify ways of utilizing both these aspects for a better outcome.

1.4 Objectives of the Work and Major Contributions

For a CNR application, the performance of task offloading decision-making is analysed based on the goals set by the user with respect to the application. Accordingly, the main objectives of our study are presented as follows:
i) **Minimizing Robotic Energy Consumption**: The primary objective we consider is to minimize the robotic energy consumption for any given application via task offloading to the cloud. We aim to find the optimal task assignment strategy for the whole application by estimating and evaluating the trade-off between the energy consumption of the robot versus offloading the tasks for remote execution [17]. Throughout the thesis, the terms task offloading, task allocation and task assignment will be used interchangeably. Moreover, since cloud networked robots have on-demand mobility, we identify key parameters (i.e., communication links and mobility) that can be utilized by the robot to further improve the offloading. Even though movement itself causes additional energy, a proper trade-off between movement energy and improved bandwidth needs to be considered during making the task offloading decisions, which may result in further reduction of energy consumption for the robot (one of our major contributions).

ii) **Minimizing Time/Delay**: Depending on the choice of the user and type of the application, reducing the application delay/response time is gradually becoming a vital issue, especially for computation intensive robotic applications. For that reason, the task completion time is another primary aspect that must be considered. By identifying the optimal task offloading decisions, the robot utilizes its resources for faster execution of tasks. Here the key is to offload the task using the best communication link, so that tasks are sent to the cloud and results are received as quickly as possible. In this way, the robot’s mobility and choice of AP also affect the decision-making performance by shortening of application time/delay. Designing both an energy and delay minimization technique allows the robot to operate with more flexibility in dynamic operations where robots may need to operate with changing objectives.

iii) **Joint Minimization of Distance and Energy**: For robotic applications, on-demand mobility helps the robots gain access to suitable APs and save energy. However,
movement itself has an additional energy requirement. So, one of our objectives in this study is to find a proper balance of energy and distance while rendering the offloading decision-making, leading to further advancement in offloading outcome.

Overall, the emphasis in this work is to improve the system performance of the CNR application via task allocation among available resources (robot and cloud) along with exploitation of key parameters (energy, time, distance). The following list outlines the major contributions of this work:

1. Develop an integrated task offloading framework for CNR applications (i.e., smart city and smart manufacturing) in order to analyse the impact of mobility and communication aspects in task offloading decisions through simulations in a smart city scenario.

2. Utilize the interdependent relation among offloading, path planning and access point (AP) selection for CNR and build a mobility-driven and communication-aware task offloading mechanism.

3. Design a 3-layer genetic algorithm-based (GA) scheme (offloading, path planning and AP selection) for smart factory maintenance application to solve the offloading optimization problem for a cloud-assisted robot.

4. Build a task offloading mechanism for a cloud networked multi-robot system and design a 4-layer GA-based decision-making scheme (task offloading, robot selection for offloading, path planning, AP selection) for smart warehouse application where multiple service robots are deployed.

In summary, the work presented in the thesis has the potential to significantly enhance performance of different CNR applications (i.e., singe, multi-robot) by integrating the key parameters (mobility and communication) in its task offloading decision process.
1.5 Organization of the Thesis

The thesis is concerned with studying the impact of mobility and communication in task offloading for different types of CNR applications and finding ways to utilize their relationship for an optimal/near optimal task offloading strategy. This helps our organization of the thesis as follows:

Chapter 2. Task Offloading Optimization for Cloud Networked Robotics

This chapter presents the prior art pertaining to different task offloading techniques in CNR as well as the implications of mobility and communication in offloading decision-making. This motivates us for a comprehensive study on the relation of communication and mobility with task offloading in the context of CNR. Later, it puts the contributions of the thesis into perspective, which is the development of a communication-aware task offloading scheme for different types of mobile CNR applications that considers a genetic algorithm to demonstrate their performance through simulation.

Chapter 3. Optimal Task Offloading Framework for Cloud Networked Robotics

In this chapter, we are concerned with the development of an integrated framework for the purpose of task offloading in CNR applications. Additionally, we present the scope of our work, which highlights two major scenarios: smart manufacturing and smart city. We initially present task offloading as an optimization problem for a single taskflow application (crowd control) in a smart city scenario, where both mobility (movement) and communication (bandwidth) is fixed. Based on this, a genetic algorithm-based scheme is designed to find the optimal task assignment decisions for each task. The initial results highlight the benefit of using the cloud in a CNR application, as it outperforms notable algorithms such as greedy, exhaustive and the all-on-robot approach. Even though
mobility and communication (bandwidth) have been considered fixed during the application, a thorough study is conducted later in the chapter to analyse the influence of such parameters in offloading decision-making. This helps us understand the combined importance of mobility and communication in task offloading for CNR applications. We also present a multi taskflow application, that further highlights how movement decisions/path planning (according to the choice of taskflow order) and its resultant communication links potentially improves offloading as well as the system performance. This motivates us to further utilize the interdependent relationship of mobility and communication with task offloading in the context of CNR as both are key parameters towards attaining a better performance for task offloading decisions.

The contents of this chapter have been published in “Journal of Sensors and Actuator Network” [AR-1] and “IEEE Global Communications Conference (Globecom 2016)” [AR-3]

Chapter 4. Communication-Aware Optimal Task Offloading for Mobile Cloud Assisted Robot

To address the benefits of mobility and communication in task offloading, we design a genetic algorithm based 3-layer decision-making scheme that takes movement decisions and AP selection as part of its decisions. Based on the relation among motion, connectivity and offloading, we present a task offloading mechanism and also propose a smart factory maintenance application for a cloud-assisted robot where the robot plans its path and selects its AP in order to facilitate the task offloading decisions. Thorough simulation analysis is then done to verify its outcome, which highlights the impact of mobility and communication on further improving the performance of offloading (with respect to our results in chapter 3). Furthermore, we also explain the proportional relation
between distance coverage and energy in order to design a weight-based offloading scheme that adjusts its objective (minimize energy, distance or both) in accordance with the robot’s battery life. Finally, we highlight another aspect of mobility-driven and communication-aware offloading when we present a recharge-based offloading scheme where the robot has the option to recharge during the application and facilitate its recharging (as long as it meets the application criteria). The complete result suggests that both mobility and communication provide improved system outcome in a cloud-assisted robotic application when considered as part of its offloading decision-making.

The contents of this chapter have been accepted for publication at “IEEE Transactions on Industrial Informatics” [AR-2] and have already been published in “IEEE Global Communications Conference (Globecom 2017)” [AR-4]

Chapter 5. Energy-Efficient Optimal Task Offloading for Cloud Networked Multi-Robot Systems

In this chapter, we consider a multirobot cloud networked system. Initially we explain the singular offloading mechanism for a multirobot system compared to a single robot system. Multirobot has more potential for a better technique of task offloading; however, the decision-making becomes more complicated. In addition to mobility and communication, a proper trade-off is required when considering the choice of offloading, as the task can be completed locally as well as on the cloud. All these considerations help formulate our joint optimization problem and eventually design a GA based 4-layer decision scheme that consists of cloud-based offloading (robot-cloud), local offloading (robot-robot), mobility (path planning) as well as communication (AP selection). For the smart factory environment, a warehouse parcel management application is considered, which proposes the idea of additional robots helping with the task offloading of a primary
robot. Thus, the workload can be shared with other robots in order achieve more energy-efficient offloading. In order to validate the outcome of our method, the GA-based scheme is investigated rigorously, and the results are compared with the previously designed/validated techniques in chapter 4, where mobility-driven and communication-aware offloading is performed without assistance from additional robots. In contrast to the previous chapter, our results here highlight the benefit of preparing an offloading framework where additional available robots help with the offloading approach. While it further complicates the decision process, the results are improved through the multi-robot dynamic in task offloading decision-making.

The contents of this chapter is currently under revision at “Computer Networks” [AR-6].

**Chapter 6. Conclusion and Future Work**

This chapter summarizes the contributions of the thesis and discusses various open problems that are associated with this work. We also present ideas for further research directions based on the findings of this thesis.
Chapter 2.

Task Offloading Optimization for Cloud Networked Robotics

This chapter provides a brief overview of the literature regarding task offloading in cloud networked robotics and related optimization studies. Hence, it serves to give the background for our work and reveals the positioning of our contributions. From the context of literature, we explain the importance of two key parameters (communication and mobility) that influence the performance of CNR. We then highlight our contributions by analysing the unique relationship among task offloading, path planning and access point (AP) selection for CNR applications (due to their on-demand mobility). The interdependent relationship among these three parameters helps design our mobility-driven and communication-aware offloading for a cloud-assisted robotic system where both movement (mobility) and communication link selection (access point) are considered as part of offloading decision-making in both single and multi-robot systems.

2.1 Optimization Techniques for Task Offloading

Mathematical optimization refers to the selection of the best element from some sets of available alternatives based on a given criterion. From the perspective of performance and QoS, the study of optimization means finding certain values of parameters that result in target achievement of a given system while meeting some constraints. In the field of engineering, there are ample studies of optimization on different topics and areas. The optimization problem is generally formulated as a non-convex quadratically constrained quadratic program, which is NP-hard in general. In this thesis, we focus on such optimization studies based on task offloading for CNR applications.
As previously mentioned, CNR is motivated from mobile cloud computing (MCC). MCC integrates the cloud computing infrastructure into the mobile environment in order to overcome the obstacles related to the performance (e.g., battery life, storage, bandwidth), security (i.e., privacy, reliability) and system environment (e.g., scalability, heterogeneity, availability) [18]. Moreover, the ubiquitous cloud resources extend the capabilities of mobile devices to improve the user experience [19] through its aforementioned traits. Since the offloading process heavily affects the integration between mobile and cloud, several studies have focused on the task offloading optimization techniques as such to attain a better quality of service (QoS).

Before 2000, the main objective for studies on offloading was to make it feasible, the reason being the limitation of wireless networks. After 2000, the emphasis moved towards the development of algorithms to identify the key offloading decisions. During this period, some studies highlighted methods that included the entire application being offloaded to the cloud by a single user, as seen in [20]. This followed studies by Chen [21] as well as Meskar et al. [22] where each task was to be offloaded entirely by an individual user. This also led to more advanced techniques with applications being partitioned into multiple tasks and the workload being shared among users as well as the cloud. Zhang et al. in [23], Wu et al. in [24] and Cuervo et al. in [25] all designed such optimization techniques for their mobile systems.

Development in virtualization then led to improved offloading infrastructure with better quality of service. Since conventional mobile systems have users that communicate with cloud servers over a long delay, ideas such as mobile edge computing (MEC) [26], cloudlets [27] and fog computing [28] were proposed to install computing resources near the base station of cellular network. Wu in [29] provided a generic offloading system with details of its components. Such frameworks provided valuable insights regarding
offloading which in-turn improved the QoS in terms of optimal task offloading. However, the most important factor for offloading performance remains to be the decision-making. Depending on the major objectives of the application (saving energy/time or saving distance/bandwidth), the decision process may vary. Henceforth, different optimization techniques have been designed for such specific cases. For example, many offloading optimization techniques focused on improving the outcome by saving energy or increasing battery life [10] [30] [31] [32]. Then, there are approaches that hastens the process by saving time [33] [34]. These techniques are developed by considering key parameters that affect offloading (such as bandwidth, server speeds, cloud access, amount of data for transmission etc.). Furthermore, depending on the type of optimization technique, offloading may happen at different granularities (task based [35], application based [31], data based [36] or at levels of method [37] etc.). In this section, we will study the different optimization techniques and their performance in more details.

As mobile cloud computing and cloud networked robotics have similarities, there is an overlap in the types of algorithms/techniques designed for offloading. For the single user case, some offloading techniques specifically emphasize on the implementation of the offloading mechanism on the mobile device (to the cloud). For example, energy-aware offloading of mobile code to the cloud infrastructure was proposed by MAUI [25]. Thinkair is another process that allows developers to migrate smartphone applications to the cloud more simply by allowing method-level offloading of computation [38].

However, both processes have limited focus on the actual optimization of offloading decisions. Besides, they concentrate on offloading the entire application or portion of it. Notwithstanding, there are studies that paid attention to offloading approaches at a “task level”. Such methods include [23] where the optimal task offloading algorithm “LARAC” was designed for a mobile application based on heuristic policy. Unfortunately, their
application considers tasks to have a linear topology, hence it is applicable to only simple scenarios. There also exist studies such as [39] by Mahmoodi et al. and [24] by Wu et al. which implement task offloading techniques for applications with task graphs that contains dependencies. In the case of our application for a cloud-assisted robot, we find most similarities to Kao et al. [34] [40] where the application task graph is represented by a direct acrylic graph (DAG). Given our proposed application type (presented in chapter 3), our application graph is similar to Fig. 2.1, hence each task in the task graph is presented as a possible offloading decision in our formulation, either to be completed on-board or in cloud. However, instead of a dynamic method presented by Kao et al., we present a GA-based evolutionary scheme for finding optimal offloading decisions.
With respect to applications with a single user, task assignments in multi-user applications tend to be more complex, which is why they require separate specific studies. In the case of multi-user MCC, initial studies focused on methods that allocated a single task to be processed by each user. This led to optimization studies such as Ren and Schaar [41] for real-time stream mining that minimized energy cost, as well as Kaewpuang et al. [42] who solved optimization models to solve resource allocation problems in the mobile cloud environment. Then there were alternative approaches that considered identifying offloading decisions for each user [22]. Some recent approaches present multi-user offloading as joint optimization problems where each mobile user has multiple independent tasks [43] and it is solved by using methods such as linear programming and stochastic modelling. Similar to the aforementioned study, we design an offloading algorithm specifically for multi-robot application in chapter 5 of our thesis and we also present offloading as a joint optimization problem for task offloading; however, we solve the problem by developing an evolutionary technique known as GA.

In comparison to the traditional optimization methods, there are many evolutionary algorithms that are designed for offloading decision-making, owing to their applicability and suitability for solving combinatorial problems in complex scenarios. Given its computational cost, our initial perception was to develop such programs for static problems. However, more studies have revealed that evolutionary algorithms can withstand dynamic scenarios and provide real-time responses in different types of scenarios. Therefore, these methodologies not only compete with the so-called “traditional methods” e.g., Artificial Neural Networks (ANNs), heuristics, machine-learning and Fuzzy Systems (FSs), but also, they outperform them for many complex problem sets.
Several researchers in the recent past have utilized different evolutionary approaches in cloud computing. Most notably, Ant Colony Optimization (ACO) has been used for dynamic job scheduling [44] [45] [46] as well as workload distribution among cloud nodes [47]. More specifically, an ACO-based offloading technique has also been presented by Bao et al. in [48], whereas an automated application offloading middleware for the ACO-based decision-making process has been presented by Golchay et al.[49]. Another well-established method is the Bee Colony Algorithm (BCA). BCA has been designed in order to conserve energy during resource utilization for cloud computing by Kansal and Chana [50]. Also scheduling algorithms in the cloud environment has also been shown to be using bee colony algorithms that aim to optimize energy efficiency while providing guaranteed optimum response delay/time [51]. Particle swarm optimization (PSO) is another variation of evolutionary computation that is being used for load scheduling [52], task scheduling [53] [54], workflow scheduling in the cloud computing environment [55] [56]. The same can be said for the Cuckoo Search Algorithm (CSA) that is also used mostly for scheduling problems [57] [58] [59], although there is evidence of a resource management problem solution [60] as well. Also some other evolutionary methods exists that are used for cloud computing models including Firefly Algorithm [61] [62], Harmony Search Algorithm [63] and memetic algorithm [64], even though they are less known. As seen in the references, most of the work with evolutionary computation tends to center on scheduling problems. Moreover, many aforementioned resource management problems studying offloading/allocation may not be suitable for multi-objective works. In order to solve task offloading optimization problems in cloud computing, there is the potential need for algorithms with the capabilities of solving multi-objective optimization problems in environments that are unstructured. Based on the examples provided above, this is where the Genetic Algorithm (GA) makes its mark.
The genetic algorithm (GA) is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics [66]. It represents an intelligent exploitation of random searches, which is used to solve optimization problems. In GA, weak and unfit species are faced with extinction by the process of natural selection whereas the strong ones have a higher possibility of passing their genes to future generations via reproduction [67]. Through an heuristic approach, the optimized solutions are obtained in GA from a number of candidate solutions [68]. Although randomized, GAs are by no means random; instead they exploit historical information to direct the search into the region of better performance within the search space. A detailed reasoning for our choice of GA is presented later in this chapter.

Similar to other evolutionary techniques, the genetic algorithm has been designed in the context of mobile cloud computing for task scheduling [69] [70] [71] [72] as well as load balancing [73] [74] [75]. Due to its scalability and adaptability, GA has been part of
hybrid methods [76] [77] [78] that are used in cloud computing for mobile users. In fact, there are many recent studies, which focused on mobile cloud computing applications for optimizing their offloading decision-making. Balakrishnan and Tham [79] worked on mapping and scheduling via code offloading, whereas Goudarzi et al. [80] introduced the idea of offloading parts of the application via a GA-based method. None of these researchers actively considered the key parameters (bandwidth, movement, network availability) that may impact the offloading performance. Abd et al. also proposed an energy-aware and fault tolerant task offloading process. Although such results are promising, they are limited to MCC only. Most importantly, it can’t be transitioned into CNR. Given the application scenario for robotics, GA is a feasible option that has the potential to perform well in the unstructured environment and outperform the traditional methods as well as other evolutionary approaches.

With respect to all the referred work, the most appropriate and relevant reference to our work is found from Kao et al., [34] [40]. This study motivated our problem formulation. But unlike their work, our domain is CNR, where robots have on-demand mobility. Therefore, mapping the task to the robot requires considerations of the robot’s location (movement) as well as available communication links (bandwidth). In our work, we have considered the impact of both parameters. The GA-based offloading approach, was inspired by Zhang et al., [65], who defined the design process for task offloading method for mobile cloud computing (as seen in Fig 2.2). Even though we follow a similar approach, the added attributes of mobility entail a multi-objective decision-making process for CNR that involves movement decisions (path planning) as well as choice of communication links (AP selection) as part of the decision-making for offloading. These unique features of robot over the mobile network leads us to the following section, which discusses different task offloading strategies specific to the cloud-aided robotic networks.
2.2 Task Offloading in Cloud Networked Robotics

As previously mentioned, mobile cloud computing and CNR have lots of similarities in terms of task offloading. However, the on-demand mobility allows robots to use path planning to access better communication links. Moreover, the types of tasks completed by robots involve actuation, which requires higher complexity and computation. All these factors open several avenues for exploration in cloud networked robotics.

Even though the research on networked robotics with additional web support can be dated back to the late 90s, it is slowly reaching its peak demand in recent times. Initial studies introduced the idea of the cloud in networked robotics such as Hu et al. [11], Wan et al. [81], Kamei et al. [82] and Guizzo [83]. This is followed by the presentation of the infrastructure and framework for offloading in CNR. Notable studies include Osunmakinde and Ramharuk [84], Bogue [85], Mohanarajah et al. [86] and Arumugam et al.[87]. Later attempts at cloud-based research mostly emphasized on different cloud computing models for robotics. For example, DAvinCi uses the Software-as-a-Service (SaaS) model for simultaneous localization and mapping (SLAM) [87]. CORE utilizes the SaaS for a distributed and scalable cloud-enabled architecture for object recognition [88]. Alternatively, Platform-as-a-Service (PaaS) is a model used by Rapyuta [89] for applications such as RoboEarth that uses the cloud for robot knowledge sharing. Robot-as-a-Service (RaaS) [90] is another alternative model that was designed to comply with common service standards for robotics operation. Depending on the choice of framework and type of application this may vary.

For collaboration between robot and cloud, there have been several studies that have focused on the different aspects of the decision-making, such as resource management [91] [92] [93], scheduling [94] [95], partitioning [96] [97] etc. Even though these studies
are beneficial, many of them may highlight on particular aspects (control, actuation, movement, communication) of robot application. For example, some robotic application-oriented studies such as small batch assembly robots [98] and robot navigation assistance [99] have proposed the concept of leveraging the cloud in multi-robot operations. Then there are also specific studies in this field that paid attention to the cloud aspects. Due to the high demand for computing resources from service providers, the computing industry has shifted towards providing different application-oriented services that can be utilized by the users. All these applications actually use the cloud in different ways. For example, projects like RoboEarth [100] provides a database for robot knowledge-sharing, whereas Xu et al. [101] introduces the idea of the virtual machine (VM) managed in cloud computing. Whaiduzzaman et al. [102] proposes a cloud service selection method for robotics based on multi-criteria decision analysis. Contrary to all these, our work utilizes cloud as a computational tool to be used on-demand for the execution of several tasks.

Another use of the cloud is as a storage repository [103] by utilizing information obtained from spatially separated robots and reproduce a user's situation. However, in our work, we prefer to concentrate on the robot-cloud interface. Hence, the cloud in our work is being used to execute on-demand task offloading rather than as a storage space. Besides, we present our offloading decisions as part of an optimization problem. Quality of service (QoS) is another aspect that is a key topic of interest in cloud robotics operations. Some recent implementations on this topic include work from Osunmakinde et al. [84] [104] where they focus on creating a framework that assists the complete network to overcome the challenge of disconnection while maintaining QoS.
Figure 2.3: Dynamic collaboration between underwater aerial vehicles and cloud from Pandey et al. [105]

In contrast, we present an integrated framework that supports the network with optimal task offloading in order to improve the QoS by considering key factors (communication and mobility) that influence the offloading decisions. We strongly believe, by combining the mobility and communication aspects with task offloading, the decision-making process will become more efficient (although at the cost of adding complexity).

Since both communication and mobility significantly influence the task offloading process (explained later in the thesis), it is worth considering these factors as part of the task offloading decision-making in the context of CNR. However, adding the movement decisions and communication link choice with offloading changes the task offloading decision process into a multi-objective optimization problem. Several optimization methods in the literature have been previously designed for single and multi-agent cloud robotics application such as Wan et al. [106] for material handling, Turnbull et al. [107] for multi-robot formation and Mohanarajah et al. [108] for 3D mapping.
Figure 2.4: Data retrieval framework of cloud robotic system from Wang et al. in [109]

As most of these approaches are application-specific, we require a more generic optimization study with a broader range. Many offloading optimization techniques exist that are adaptable specifically for optimal allocation in CNR. Some of them are: particle swarm optimization [110] [111], ant colony optimization [112], the greedy algorithm [113], and the dynamic algorithm [114]. Among the studies found in the literature, the offloading problem for robotics that mostly resembles our work is Pandey et al.’s [105] who studied task mapping in an underwater mobile sensor network by implementing a dynamic and reliable collaboration between an autonomous underwater vehicle (AUV) network and the cloud data centre. Wang et al. in [109] is another relevant work that presented a real-time multi-sensor data retrieval framework (see Fig 2.4) for multi-agent cloud robotics systems where they presented allocation and scheduling for a 3D environment mapping. Even though our application scope is broader, both these studies
present task offloading as an optimization problem, which is similar to ours. Unlike their methodology though, we present a GA-based scheme and our framework focuses on the task offloading decisions rather than resource allocation in the cloud.

While classical problems such as task offloading, scheduling and load-balancing have already been addressed by the Computation Intelligence (CI) community, using techniques such as evolutionary computation for optimization has the potential to deliver improved performance, especially in complex scenarios. As stated previously, the genetic algorithm is one such method that has significant advantages over traditional and other evolutionary techniques found in the literature. For evolutionary robotics, GA is a machine learning approach that has been traditionally used to optimize the control policy of a robot [115]. It is implemented in applications to rapidly locate “satisficing” solutions when sufficient a priori knowledge is unavailable. So, GA is especially useful in robotics for unstructured problem scenarios with multiple objectives in unknown environments.

![Diagram](image.png)

Figure 2.5: An energy sensitive GA-based task offloading mechanism proposed by Guo et al. in [116]
As found in the literature, individual studies of GA in classical machine-learning problems require adaptive learning without significant domain knowledge for finding the solutions. Task offloading in cloud robotics is an example of such a topic where a GA-based approach is suitable given the nature of the problem set. However, most of the GA-based approaches in literature are exclusive to mobile cloud computing applications/scenarios. Given the on-demand mobility of the robot, three key parameters (offloading, movement and communication) are interdependent for the robot, which broadens the problem space as well as increases the complexity of the decision process.

In such a case, GA is an ideal approach for our application domain of CNR. The most relevant and recent example of the GA-based method in cloud robotics is found from Guo et al. [116] who presented an energy-sensitive GA strategy to offload tasks to the cloud center. Even though this approach improved the computing ability and execution efficiency for cloud robotic networks (Fig. 2.5), but their work only focused on the mapping of tasks (for offloading decision-making) to a GA approach. On the other hand, our problem in this thesis considers both movement (path planning) and communication (AP selection) aspects as variables along with task offloading, which makes the problem multi-objective. Given the increased complexity, we map all these three sets of decisions to a modified GA based approach that has the capability of solving such multifaceted problems.

With the rapid increase in technology, the complexity of handling dynamic and multifunctioning systems is exponentially increasing day-by-day because of factors such as dependencies among parameters, difficulty to map, interconnections etc. In order to avoid situations where certain aspects of development may become “intractable” due to constant progress and evolution in response to progressive conditions and demands, it is of utmost importance to prepare comprehensive techniques for system modelling so that
it can constantly deal with various dynamic changes and high levels of complications that may arise. Therefore, more interdependent parameters are integrated to formulate and solve joint optimization problems (such as our study) where the algorithm is trained to be more rigid and driven towards an area of optimal result with high probability in an efficient way.

In order to facilitate all these requirements, we aim to design a modified GA-based scheme that has the capacity to deal with complexities of a multi-objective optimization problem and perform in an efficient manner. Accordingly, we need to thoroughly study the two exclusive and interconnected parameters (i.e., mobility and communication) and their relation to task offloading for CNR. Once this has been established, we further study the ways to integrate all three as part of the decision-making. In the following section, we will specifically highlight these abovementioned issues.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Cellular</th>
<th>Wi-Fi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delay</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Energy-Efficiency</strong></td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Figure 2.6: List of differences between cellular network and WiFi for cloud networked robotics (reproduced from [29])
2.3 Implications of Communication and Mobility

For CNR, the ability to move on-demand means the robot can potentially make its movement decisions and choose its link selection to accommodate/influence its offloading decision-making. This is unlike the MCC systems, where both mobility and communication aspects tend to be passive (user-dependent). Consequently, it is important to understand the impact of communication as well as mobility on task offloading with respect to different systems. The following section provides a detailed description of the studies undertaken so far, which then motivates our novel approach in the CNR application space.

2.3.1 Communication aspects

As previously mentioned, task offloading to the cloud is an emerging trend in distributed computing for networked robotics. Motivated by recent advances in wireless communication and IoT-enabled devices, the CNR architecture distributes computation, communication and storage between the cloud and robot. In order to boost the network capacity and meet the real-time demands of such applications, more attention is currently being provided on the communication aspect of robot-cloud collaboration. While tasks are executed on the robot, the computation intensive tasks consume high energy and latency [117]. To address the computation and performance issue, tasks can be wirelessly offloaded to the cloud servers, to be completed on the much more advanced and highly equipped virtual machines. However, offloading the task to the cloud incurs additional latency and energy for the cloud-robot communication. The key factor here is the bandwidth of the communication link. A fast communication link (high bandwidth) shortens latency for cloud communication and results in less energy consumption for the robot; whereas, a slow communication link (low bandwidth) may result in higher energy
consumption, which may make it infeasible/non-beneficial to offload the task. Robots, similar to mobile devices, have multiple interfaces such as 2G, 3G, 4G, LTE, EDGE for data transfer over the cellular network. They also have the WiFi network that varies in delay, energy cost and availability. Considering such factors, Wu [29] presented a list of differences between cellular networks and WiFi for mobile users such as robots, as presented in Fig. 2.6. This suggests that offloading clearly provides differences in performance based on the proper selection of network, since each of them has different modes of communication. In our work, we have chosen the WiFi network, as our focus is to be energy-efficient, which is one of the benefits of WiFi over the cellular network (as seen in Fig. 2.6). For the WiFi network though, the communication link/gateway to connect the robot and cloud plays a key role.

Therefore, many researchers studied the communication link between the user and cloud infrastructure in order to improve the task offloading and decision-making. For example, Wolski et al. [118] proposed the idea to consider bandwidth as a criterion for offloading decisions, which was later followed by Pawar et al. [119] for mobile cloud computing. Some studies such as Barbera et al. [120], and Mehmeti and Spyropoulos [121] took the availability of communication links into consideration and proposed their methods based on this. Furthermore, there are studies that focused on the selection of the communication channel for offloading decisions. These include studies by Zhang et al. [122] whose objective was to minimize energy for real-time video applications under dynamic wireless network channels. Another example is Wu et al. [123] who tried to indirectly select the communication link through cloud path selection and Han et al. [124] who considered user numbers to select the communication link with minimized traffic for faster communication/offloading. At the same time, several optimization studies [125] [126] have been proposed that proved that the selection of communication link for
offloading brings about superior system outcome through improved throughput for user and cloud-based communication.

For mobile users, several studies also highlighted the Internet gateways as possible ways of selecting channels. As users in this context are making headway, this opens the possibility of selecting the best channel to suit its offloading requirements. Most prominent is the selection of access points (APs) that have the potential of significantly improving performance for task offloading to the cloud. For studies related to access points, Bulut and Szymanski [127] proposed an AP deployment scheme with the aim of achieving a large number of offloaded total data traffic volume from all users, at the same time increasing user satisfaction. The selection of AP was presented as part of an optimization study by Liu et al. [128], where an algorithm was proposed based on the Analytic Hierarchy Process (AHP), which led to superior system performance. Instead of single optimization, as in their work, we propose a joint optimization problem in our thesis that presents AP selection along with offloading as part of decision-making. Similar to our work, Chen et al. [129] also proposed a joint optimization problem where offloading decisions were optimized along with allocation of computation and communication resources in order to minimize energy consumption. Although our objective is the same, our application scenario involves a robotic network that has unique features of on-demand mobility that affects the communication aspect of offloading (discussed later in the thesis).

As previously mentioned, in comparison to mobile users, networked robotics have the potential to further utilize the communication aspect for offloading to the cloud. The on-demand mobility allows robots to choose their locations for offloading, based on their selection of the network gateway (AP). Several current studies on communication for cloud robotics have not utilized this concept yet. For example, Dhiyanesh [130] proposed
communication protocols and present cloud robotics architecture for dynamic resource allocation in a cloud robotics system. However, it is a single objective problem and does not consider the decision-making along with bandwidth. We, on the other hand, propose to integrate the network selection (through choice of AP) with the offloading decisions, which would make it a joint optimization problem. Chen et al. [91] proposed a workflow allocation algorithm for joint optimization which includes communication and is QoS-aware. However, unlike them, we consider the additional aspect of on-demand movement that makes our framework mobility-driven and communication-aware. Although, Wang and Meng [131] have proposed a bandwidth allocation framework for improved offloading in cloud robotics, they presented a game theory method. In our work, we have proposed a genetic algorithm-based method that is highly suitable for multi-objective problems such as ours. Another joint optimization with which we have similarities was designed by Chen et al. [43] who proposed joint optimization of communication resources and offloading decisions. However, their study focused on MCC, whereas our study was for the domain of CNR. Moreover, we added movement decisions along with communication and offloading for a more complex joint optimization study.

In summary, this section presents the details of one of the key factors in making offloading decisions, which is communication. We believe that choice of communication link plays a critical role in task offloading and initiates faster communication with the cloud. However, it is driven by the ability of the robot “to reach” the appropriate locations in order to find suitable APs for cloud communication. Since on-demand mobility allows robots to gain access to such benefits, it is obvious that communication and mobility are interconnected for offloading decision-making, which is why a thorough study is required to study the impact of mobility on task offloading. This introduces the next section on the aspects of robot’s mobility relevant to task offloading.
2.3.2 Mobility aspects

Similar to communication, mobility plays an important role in task offloading. For mobile cloud computing, mobility mainly points to the user movement and the resulting bandwidth that is used for cloud communication. As a result, many researchers have considered user mobility for offloading in mobile cloud computing. For example, Rahimi et al. [132] proposed a framework for mobility-aware service allocation where the application was modelled as location-time workflows of tasks and mobility patterns translated to a mobile service usage pattern. Another relevant work was produced by Wang et al. [133] who investigated the problem of mobility-assisted opportunistic computation offloading by exploiting the contact patterns regulated by these devices' mobility. This later helped with understanding the amount of computation that requires offloading. Xia et al. [134] considered an online location-aware offloading for a two-tiered system (consisting of cloudlet and remote cloud) with the view of attaining/achieving fair-share use of cloudlet, cloud and mobile users. These are a few of the notable studies from the literature that have set the precedence of using mobility for different types of offloading in mobile cloud computing. Most recently, Shi et al. [135] proposed a mobility-aware offloading decision method for distributed mobile cloud computing for cloudlet reliability estimation and used an integer encoding-based adaptive genetic algorithm for the offloading decision. Moreover, Wang et al. [136] and Chen et al. [137] proposed mobility-aware task assignment/offloading for different applications of mobile cloud computing. However, unlike all these studies, our application space is cloud robotics, where robots make on-demand movement. Therefore, instead of relying on user mobility patterns or making mobility predications, the robot is able to actually plan its path and accommodate offloading. Although our work also uses GA (same as
[135]), contrary to them we present our problem as a joint optimization which includes the consideration of mobility along with offloading and communication.

Several instances on literature have previously attempted to study hybrid methods for offloading that included mobility. Most prominently, Akter and Zohra [138] proposed a QoS and mobility-aware resource allocation architecture for remote code execution in order to achieve efficient latency and reliability. More recently, a joint scheme for mobility-aware caching and computation offloading was proposed by Chen et al. [137] with the view to improve the system performance in terms of energy cost. For their communication purposes, they considered the 5G ultra-dense cellular networks. Despite bearing resemblance to both studies, our work differs in its approach, which is to consider communication and mobility as part of a task offloading decision-making problem instead of QoS and caching.

For studies related to task offloading decision-making, some notable work can be found from Wang et al. [136] who formally modelled the problem of task assignment in mobile edge computing and proposed a lightweight algorithm for mobility-aware task assignment. Another relevant work on task offloading decisions is from Hridita et al. [139] who formulated a task allocation optimization problem by considering the mobility of the user that affected connectivity between mobile and the cloud. Finally Lee and Shin [140] studied the users’ mobility to get hints on upcoming changes for network connectivity and designed an offloading decision-making technique based on the mobility model. Although all the work is in the same field of research, our work has more similarities to the latter example. As the robot has unique yet complex features of on-demand mobility, all the movement related decisions is planned by the user (robot) while accommodating offloading along with the communication link choice.
For consideration of mobility, the main upgrade for CNR over MCC is its on-demand ability. The ability of on-demand movement to gain access to a better network produces further improvement in offloading. Even though many studies have tried to map mobile cloud computing to networked robotics framework (e.g., Ma et al., in [141], Burski and Garbacz in [142] etc.), they haven’t put emphasis on the task offloading decision-making with the perspective of adding movement to it. We have actively proposed the idea to make movement decisions based on suitable offloading decisions, which essentially points to path planning to accommodate offloading decisions. Several robotic studies in the literature have previously presented path planning algorithms [143] [144] that solely focused on movement decisions. In comparison, our path planning/movement decisions are part of a multi-objective decision-making that mainly benefits the task offloading decisions for CNR applications.

To summarise, our study is one of the first attempts to jointly consider mobility, communication and task offloading as a multi-objective optimization problem for CNR which is then solved by designing a modified genetic algorithm with multi-layer decision-making scheme.

2.4 Reasoning for the Genetic Algorithm Approach

Over the years, there has been numerous works on different types of optimization methods. However, GA in particular became popular through the work of John Holland in the early 1970s [145]. We use the genetic algorithm because it is a widely recognized global optimization algorithm that is used in many fields because of its high efficiency and impressive stability [65] [146]. In a traditional optimization problem, the aim is to minimize/maximize the objective function \( f(x) \) over a given space \( X \) of arbitrary dimension. For brute force methods, it may be infeasible to examine all possible \( x \in X \) to
determine whether the result is costly. On the contrary, the heuristic approach of GA is the ideal choice for identifying the input space $X$ for optimal solution without checking all the possible combinations [147]. One of the key features of GA is its ability to find solutions in unstructured problem sets. Given the scope of our research, this method will be suitable for current work as well as possible future modifications.

A clear distinction with other evolutionary algorithms such as particle swarm optimization (PSO) [148], bee optimization [149], ant colony optimization [150] is found with regards to joint optimization (similar to the problem set in our study). GA is initially a discrete technique suitable for combinatorial problems like ours. In contrast, PSO/bee/ant algorithms tend to be continuous methods that may perform less efficiently for combinatorial problems.

The GA technique also has a higher variability of increasing probability to find better results, due to the implementation of steps such as crossover and mutation (discussed later in the following chapters), which makes the population more diverse and thus more immune from getting stuck in a local optima. In theory, the diversity also aids the algorithm to be faster in reaching close to the global optima (contrary to an exhaustive search) since it allows the algorithm to explore the solution space much faster.

Another key advantage of GA is found in cases when the objective function is not smooth but is rather noisy or stochastic. In such situations, the derivative methods don’t always hold up in performance. Since GA doesn’t require any information about the structure of the objective function in advance, it has a high probability of dealing with uncertainty and producing better results.

Finally, GA has the ability to effectively deal with dynamic scenarios [151] since it is more adaptable and has fewer built-in assumptions that potentially constrains the problem
set from finding optimal/near optimal solutions. Although GA itself is a working method, it has the versatility to be suited to different problem scenarios, which motivates our scheme in this thesis.

Given the complexity of our multi-objective optimization problem in this thesis, a classical GA approach would have considerably higher overhead and lower efficiency. In order to overcome this issue, we have modified the GA scheme in several stages according to the needs of the application. Based on all these considerations, GA is assumed to be the ideal choice for a task offloading algorithm in our application scenario. The following section presents an overall description of our contributions made in this thesis and its positioning with respect to the current literature and application space of cloud robotics.

2.5 Positioning of the Contributions

As per section 2.3, there are ample examples in literature that suggests that mobility and communication individually play major roles in task offloading performance. However, most of these studies have been from the point of view of mobile cloud computing. As the robot moves on-demand, it significantly changes the relationship among task offloading, mobility and communication. Due to this exclusive feature, task offloading decision-making in cloud networked robotics is a distinct area and therefore deserves a separate study.

The relationship between task offloading, path planning and AP selection for CNR is depicted in Fig. 2.7. As previously mentioned, task offloading is only beneficial when transfer of a task to cloud can compensate for the robot’s local computation. If \( E_{\text{robot}} > E_{\text{cloud}} + E_{\text{transfer}} \), then task offloading to the cloud would save energy. Here \( E_{\text{transfer}} \)
means communication energy between the robot and the cloud. Since, $E_{\text{Transfer}} = Power_{\text{robot}} \times data/bandwidth$, it indicates that bandwidth may determine the performance of task offloading. However, bandwidth is dependent on location as well as choice of access point (AP). A closer distance to the communication link (AP) helps to choose suitable APs for increased bandwidth (↑), which decreases communication energy $E_{\text{Transfer}}$ (↓), thus increasing the possibility of task offloading (↑). At the same time, the choice of a suitable AP may also mean the robot has to plan its path accordingly, so as to gain access to the communication links with higher bandwidth (↑) for lower $E_{\text{Transfer}}$ (↓) and increased possibility of offloading (↑). Furthermore, offloading decisions also dictates which path the robot takes during movement as well as which AP it chooses for cloud communication. Altogether, it suggests that offloading, AP selection (communication) and path planning (mobility) are interdependent on each other, which increases the complexity in task offloading decision-making for robotics. Therefore, these three factors need to be jointly considered as part of a decision set in order to achieve an improved system performance. Based on this, we study novel ways to design task offloading decision-making strategies for different types of CNR systems.

Figure 2.7: Relationship among task offloading, path planning and AP selection
2.5.1 An integrated task offloading framework for CNR applications

In order to analyse the impact of mobility and communication aspects of task offloading, a task offloading framework is required that is capable of handling such layers of complexities. Hence, our proposed framework integrates robotic network, cloud infrastructure and wireless sensor networks (WSN) in the context of CNR applications where the combination of sensing (WSN), actuation (robot) and computation support (cloud) will provide different autonomous services. Among several possible scenarios, we choose two representative applications: smart manufacturing and smart city. The proposed framework also allows us to analyse the impact of mobility and communication through simulation, for which task offloading is presented as an optimization problem. A GA based task offloading scheme is designed to identify the optimal offloading decisions for each task where initially the bandwidth (communication) and movement locations (mobility) are considered fixed. Both bandwidth and movement locations are varied later on to analyse their impact on the offloading decisions. Finally, optimization for a multi-taskflow is formulated with the simulation results helping us to comprehend the joint influence of bandwidth and path planning on offloading decisions as well as verify the unique relationship among the three aforementioned parameters in the context of CNR (explained at the beginning of the section). This in turn motivates our following work to utilize mobility and communication aspects in designing novel task offloading decision-making strategies for different types of CNR systems.

2.5.2 Mobility-driven and communication-aware offloading for cloud-aided robot

To exploit this relationship, all three parameters are jointly considered in the context of the cloud robotics applications. In robotics, “path planning” is concerned with the problem of optimal robot movement between multiple points [152], that involves target-oriented
decision-making. The AP selection, on the other hand, directly relates to bandwidth assessment where communication channels are not dedicated to the robots, so the bandwidth is shared with other users in the network. It means the robot’s share of bandwidth is less than the maximum offered by a given AP. This is why the proliferation of wireless access technologies offer users the possibility to choose among multiple networks (via AP selection) [153] in order to achieve the best connectivity/bandwidth. Individually, path planning and AP selection have been separately considered for offloading decision-making, mostly for MCC. However, unlike the mobile networks, a robot moves on-demand, so it can jointly plan its path and select its AP for gaining access to suitable communication links, thus making offloading more efficient. In this way, integrating path planning and AP selection with task offloading for CNR is a novel approach to improve the system performance by saving the energy. In this case, path planning represents the robot’s ability to make movement decisions in order to accommodate offloading, while network connectivity implies the robot has the option to choose its communication link for offloading. Alternatively, the robot’s task offloading decision dictates the path the robot will move through. At the same time, offloading decisions may force the robot to move to certain locations that are close to APs.

In order to accommodate all three parameters, we design a GA-based method with 3-layer decision set (offloading, path planning and AP selection). Even though movement or communication has already been considered in making offloading decisions, to date, no one has utilized the interdependent relationship among offloading, AP selection (communication) and path planning (mobility) for a joint study. Given their bi-directional relationship, our novelty lies in the joint optimization approach of a communication-aware and mobility-driven task offloading approach that jointly considers all these three parameters as variables for decision-making.
Figure 2.8: Simple structure of a multi-robot cloud networked robotic system

2.5.3 Energy-efficient task offloading for multi-robot cloud networked system

For a multi-robot cloud networked system, task offloading is more complex due to the inclusion of additional robots. In comparison with a single robot, all the robots can offload tasks in this system. Consequently, local coordination among robots is also required. This further complicates the scenario. However, the addition of a robot also presents an opportunity to further utilize the system components to improve offloading. Previous studies in the literature have generally focused on multi-robot coordination [154] [155] for different types of applications (e.g., rescue mission [156], disaster support [157], navigation [158] etc.). As for cloud-based approaches, examples has been few and far between (e.g., cloud-assisted negotiation technique with industrial robots [159], localization through deep learning with cloud support [160], development of a novel
green software evaluation model for energy minimization [161], context aware offloading [106] etc.). Contrary to all these methods, our study concentrates on mobility-driven and communication-aware task offloading for multi-robots that follows a mesh model. As seen in Fig. 2.8, we consider the following two factors: (a) robot-robot communication for local offloading and (b) robot-cloud communication for task offloading to the cloud. By balancing the workload between cloud and the multiple robots, we further increase the efficiency of the system. At the same time, allowing the available robots to help the primary robot with offloading potentially results in superior offloading performance. Hence in our study, we present a task offloading problem for a multi-robot cloud networked system as a joint optimization study and solve it by using a GA based 4-layer decision-making approach that considers offloading, selection of robot, path planning and AP selection which further improves the system outcome.

As GA-based technique has the aptitude to successfully handle multi-objective problems, we design our multi-layer GA-based scheme for offloading decision-making and modify it accordingly for single and multi-robot operations. In conclusion, the GA-based approach for different CNR applications integrates offloading, mobility and communication aspects and improves the whole system performance.
Chapter 3.

Optimal Task Offloading Framework for Cloud Networked Robotics

An integrated framework for task offloading in CNR application that consists of IoT enabled sensors, cloud and the robot is designed in this chapter. Thus, different autonomous applications can run in both smart city and smart manufacturing scenarios. We present study cases as well as the scope of our work. A detailed simulation providing a near-optimal solution is shown to highlight the benefit of task offloading via our proposed GA-based scheme. Different from the traditional approach of mobile cloud computing, in our framework we analyse the impact of mobility and communication aspects on task offloading decision-making. This helps us identify the interdependent relationship among task offloading, path planning (mobility) and AP selection (communication) for a CNR system and how this can be utilized to improve their performance.

3.1 Introduction

Cloud robotics is an emerging paradigm that enables autonomous robotic agents to communicate and collaborate with a cloud computing infrastructure. It complements the Internet of Things (IoT) by creating an expanded network where robots offload data-intensive computation to the ubiquitous cloud to ensure quality of service (QoS). However, offloading for robots is significantly complex due to their unique characteristics of mobility, skill-learning, data collection, and decision-making capabilities. Having emerged as an extension of mobile cloud computing, the concept of task offloading in robotics has been extensively studied in the literature (as highlighted in chapter 2). However, contrary to the traditional approach (as in the case of MCC), our work focuses
Figure 3.1: Different cloud robotics model (Reproduced from Hu et al. [11])

on the unique features of networked robotics (on-demand mobility to gain access to the superior communication links to cloud) that is beneficial via their offloading frameworks for improving task offloading decision process in different types of CNR applications.

In the recent past, several cloud robotics frameworks have been proposed based on the type of applications. Hu et al. [11] introduced such cloud-robot models (i.e., peer-based, proxy-based, clone-based), each of which has its advantages and disadvantages. However, no detail was provided on the applications where it would be suitable. Doriya et al. [162] reviewed several cloud robotics frameworks specifically for the purpose of solving simultaneous localization and mapping (SLAM) problems. Osummakinde and Ramharuk [84] developed a survivable cloud-robot framework that is suitable for heterogeneous environments. Particularly, the latter presented the concept of a resource allocation paradigm that is enabled by the proposed framework. However, none of them put efforts on designing a framework from the point of view of task offloading. In fact, Guo et al. [116] is the first recent reference in that regards. He proposed an energy-sensitive task offloading model that enables computation offloading for robotic
applications to be deployed in the field for the purpose of exploring, monitoring, and giving feedback to harsh environments. In contrast, we believe the merger of IoT and cloud robotics has brought the scope for more integrated systems where wireless sensors with their deep learning ability communicate with cloud-aided robots for more complex autonomous applications. The key factor here is to develop a sophisticated framework that accommodates task offloading for robots as well as attain the benefit of the robot’s on-demand mobility to further improve on it later on.

3.2 System Architecture for Cloud Networked Robotics

The different available cloud computing models that incorporate machine to cloud (M2C) interaction/communication in CNR are mentioned by Hu et al. in [11] (reproduced in Fig. 3.1). As each model has its own pros and cons, we consider various factors such as QoS, adaptability, interoperability and scalability to inform our design choice. In this section, an integrated cloud networked robotics framework is proposed in order to realize both a smart city and smart manufacturing vision while taking into consideration its various complexities. Specifically, we present the components of an integrated framework for robot task offloading. With the recent inclusion of IoT, a pool of heterogeneous sensors is deployed throughout smart infrastructure that detects critical events and monitors physical magnitudes in order to develop a common operating picture (COP) [163]. With the unified framework of sensing and cloud computing resources, the biggest drawback lies in its lack of mobility and actuation. This is why the next logical step for improvement in this context is the introduction of robots. The robotic network is a complementary addition to the current static IoT devices and has the potential to be a central ICT component of the smart system. It adds an actuating dimension to complement the wireless network of sensors by interacting with the environment. As robotic agents are able to perform mobile and interactive services, they may be deployed
This integration of machine learning in IoT-enabled sensors and computation support from the cloud infrastructure motivates our four-tier task offloading framework for a cloud networked robotic system (Fig. 3.2). The components of this framework are: i) physical layer consisting of robots and sensors, ii) network layer with APs, iii) cloud infrastructure and iv) supervisory control centre overviewing the application. This highly smart and flexible framework is self-organized and reconfigurable in nature. More details on the components of the framework are provided as follows:
i) **Physical Layer:** The physical layer contains two key components: wireless sensor network and robotic agents. The wireless sensor network (WSN) consists of low-cost and limited-energy smart sensor devices embedded in machines that communicate with each other and collect raw data as well as analyse them for necessary application-specific tasks. The machine learning in sensors provide directive information to the robot through cloud by using APs. Based on this, several analytical computations are done to detect possible events which guide the robots to visit locations and complete service-based tasks.

The robotic agent is the lynchpin that connects the other components (i.e., cloud and sensors) of this framework, performs actuation and helps complete the application taskflow. Depending on the type of application, the robots use their ability to reduce their workload by offloading tasks to the cloud VMs for additional support. Robots may also get guidance provided by the supervisory control centre through the cloud as well. Therefore, several key allocation-based decisions are made to share the tasks among the available resources. Additionally, as robots possess the distinctive attribute of on-demand mobility, they plan their route accordingly in order to choose the suitable communication platform for the purpose of accessing the cloud while moving. This further improves the offloading process. Finally, for multi-robot systems, the total set of robots are also able to form their own “local ad-hoc network” to communicate and share information with each other as well as assist other robots with sending tasks to the cloud. Thus, the framework allows robots to utilize their surrounding local resources (robots) and cloud-based resources (virtual machine) to intelligently share their workload as part of a fluid communication model.

ii) **Network Layer:** The network layer consists of access points (AP) that enable the robotic and sensor network to communicate with each other. It also bridges the gap between cloud services and the physical layer components for data collection and
uploading. In this context, the AP is defined as a smart device with Internet capabilities that helps the robot access the infrastructure of the cloud. As there are multiple APs, the robots have the option to gain different stream rates for communication depending on their location and choice of AP, in accordance with protocol IEEE 802.11 WLANs. This means the bandwidth available to a robot may vary depending on the location from where it offloads tasks or the robot that is offloading or even the AP it is selecting. Thus, the tangible network layer enables the in-tangible information to flow freely by integrating physical components and information entities. More details about the communication model and bandwidth assessment for task offloading is provided later in the thesis.

**iii) Cloud Infrastructure Layer:** Foster et al. [164] state that the emergence of cloud computing in recent times has provided redundant, inexpensive, and scalable resources on demand to meet challenging and dynamic system requirements. In the context of our work, the cloud infrastructure refers to hardware and software components such as servers, storage and virtualization software that are needed to support the computing requirements of the application. It comprises an abstraction layer that virtualizes resources and logically presents them to users through application program interfaces and API-enabled command-line or graphical interfaces. The organization of the cloud typically contains virtual machines (VM) with shared power that provide the required functionality to execute the entire high-density operating systems. At the same time, the cloud infrastructure also requires massive computational capacity to handle various unpredictable and complex user (robot) demands. Some of the notable cloud service providers currently available that perform such services are: Microsoft Azure [165], Google App Engine [166], Amazon Web Services (AWS) [167], Mendix [168] etc. In the context of this thesis, we refer to the cloud infrastructure mainly as a virtual machine (VM) that virtualizes computing resources as the back-end components and perform on-
demand computational support (for offloaded task), data storage (data collected from sensors) and analytics (decision-making, verification) etc.

iv) Supervisory Control Layer: The supervisory control layer allows networked robots to be guided/monitored by humans remotely through the cloud infrastructure. Here the information collected from sensors and action reports performed by robots are passed on to the cloud and made available for users to monitor through control terminals. As a result, the physical layer communicates with users/engineers in remote locations when required. In addition, possible big data analytics also provide various statistical results to the users for the purpose of supervisory control and the users verify/adjust system configurations according to the needs of changing application condition. Particularly in large-scale operations (e.g., smart city or smart manufacturing etc.), this two-way communication allows remotely located engineers to potentially monitor robots and maintain performance of applications (if required).

3.3 Application Overview: Smart Manufacturing and Smart City

Through current efforts of IoT in a smart environment with the inclusion of cloud-enabled robots, the opportunity for innovation arises through the integration of networked robotic systems, IoT-enabled sensors and the cloud infrastructure for intelligent perception and on-demand shared resources [169]. By leveraging the cloud facilities, an autonomous robot would enhance its computational resources in order to perform more difficult yet beneficial service actions. A number of specific application domains have been identified that utilize this unified infrastructure for service operations in health services (robotic surgeries), tourism (guide robots), security services (patrol robots, crowd control robots), transport services (smart traffic police), emergency management (fire-fighting robots), factory operations (maintenance robots) etc. Among all these
possibilities we choose two representative domains for our proposed framework that suits our application scope: smart manufacturing and smart city.

### 3.3.1 Smart manufacturing

Following the first three revolutions of “Mechanization”, “Mass Production” and “Digitization”, the industrial sector is currently going through a fourth revolution in which emerging autonomous technologies are transforming traditional factories into smart factories of the future [170]. This innovation is driven by the integration of cyber physical systems and IoT-empowered wireless sensor devices that connect over networks and communicate with each other for data-sharing and automated processing of operations that start from the production line all the way to marketing. Thus the proposed system now provides a method for intelligent perception and on-demand usage of shared resources [169] to reconcile conventional problems that have plagued factory workplaces for centuries. As design and operation in industrial operations involve numerous varieties of decision-making [171], the integration of cloud computing (CC) and wireless sensor networks (WSN) has resulted in increased efficiency for environmental monitoring, improved supply chains via data acquisition, reduced waste and more safety and speed. Concurrently, robotics has also made a significant mark in applications in the industrial realm. The employment of robotics in the industrial realm can be traced back to the early 90s, where the approach initially started with teleoperation. Later upgraded to automation and with time, it has now reached the age of cloud robotics. Similar to IoT, cloud computing in robotics enhances the operations of robots via: (a) computation support, (b) storage, (c) robotic learning, and (d) crowdsourcing [13]. Therefore, integrating IoT in smart factories with the recent inclusion of robotics and cloud computing infrastructure, has led to a vast range of automated industrial applications [169]. Prime examples in
Industry 4.0 applications include: material handling [106], assembly line [98], warehouse process maintenance [172], and cooperative navigation [158] etc.

Nowadays, all the industrial (Industry 4.0) applications require autonomous systems with the ability to provide automated and customized services that deals with personalized consumption demands. Hence the implementation of cloud networked robotic systems presents itself as the perfect fit. Later in this thesis, we will focus on two specific applications of CNR. One is factory maintenance that requires robots to maintain operations and check parameters in a remotely located factory. Another is a multi-robot system where the primary robot completes the operation of a warehouse parcel management with aid from additional nearby and available robots. Both scenarios match the scope of our work and fall within the domain of smart manufacturing. In fact, our case study of smart manufacturing is the perfect example of CNR where the framework utilizes the wireless sensor network (WSN) and the cloud infrastructure to patrol different areas and control factory applications in an optimal way.

3.3.2 Smart city

One of the most promising avenues for implementation of cloud robotics is in a smart city paradigm. Jin et al. [173] define this as a city that utilizes information and communication technologies to make services and monitoring more aware, interactive and efficient. A smart city interconnects the physical, ICT, social, and the business infrastructure to leverage the collective intelligence of the city [174] [175]. Accordingly, the integrated cloud networked robotics framework provides numerous services to its citizens. This has led to recent studies that are motivated by numerous opportunities for novel applications in this application domain. A city-wide wireless sewer sensor network by Jeong et al. [176], an Urban Automation Network (UANs) by Gomez and Paradells
and a fully remote-controlled street lighting aisle of lamp posts by Leccese et al. [178] are examples of such approaches. Our difference is that our application involves a WSN and robotic network being used for surveillance; hence the scope is different.

Calavia et al. in [179] presents a proposal of an intelligent video surveillance system that is designed to minimize video processing and transmission in a camera network deployed in smart cities. However, their application lacks movement, which is not the case in our application due to the involvement of cloud-aided robots. Although the cloud-based approach for a smart city robotic system has already been studied by Ermacora et al. [180] [181], their application focuses on the implementations of Unmanned Aerial Vehicles (UAVs). We, on the other hand, pay attention to applications with automated ground robots, as presented in the next section.

Figure 3.3: A smart city cloud robotics application of crowd control [182]
In this section, we will formulate our task offloading optimization problem for a smart city application and analyse the impact of mobility and communication aspects on offloading through simulation. As smart city is a multi-dimensional concept, it is able to
evolve and adapt with upcoming technologies. The application we present is of “cloud robotic crowd control” in a smart city scenario [183] (e.g., station, park, building, or stadium) which involves a camera network (for surveillance), a robotic agent (patrol mode) and a cloud infrastructure (computation support). The complete operation can be divided into three integral stages/events that require the robot to visit three distinct locations. A 20-node taskflow in section 3.4 represents the different aspects of the operation where the performance of the system will depend on the completion of all these tasks.

i) Identification Event: This initial stage of the application is a merger of surveillance by camera sensor networks and the robotic agent. The camera WSN collects surveillance data. Upon this data, heavy analysis is done by the camera to detect and identify any person of interest who may be a threat or require assistance. This triggers the first-on-scene incident. The location data of the person is wirelessly forwarded to the robotic agent that is positioned within the range of the WSN. Based on this information, the robotic agent then devises a path plan to move to that location. As seen in Fig. 3.3, path planning involves heavy computation tasks such as path analysis and collision detection.

ii) Investigation Event: The next step of the application requires the robot performing some investigative tasks. Upon arrival, the robot identifies the correct person to interact with via a face match with the agent-mounted camera. Then the situation is assessed by capturing and analysing the gesture recognition, now in more detail. The agent also begins verbal communications with the person to detect speech patterns. With these analyses complete, the system attempts to determine the best course of action: continue questioning, contact the appropriate authorities, or guide the person to a nearby safe location. The investigation event does not require much movement; however, it consists
of several complex tasks to be completed in a timely manner in order to make proper
decisions for the next stages.

iii) **Guide Event:** Another major aspect of the robotic crowd control system is the
service provided by the robotic agents in the guide event. When the robotic agent needs
to escort/guide a person of interest to a given location, then this event is triggered. At that
point, the robot will perform the path planning operation and move to the intended
location while guiding the person and avoiding obstacles. The guide event is a unique
feature of the robot as it involves movement as well as interaction. All of these tasks
involve complex computation in order to provide the necessary service.

For the application, all the components is presented through a three-tier architecture
(Fig. 3.3). Firstly, the sensor network is represented by the camera network that is
wirelessly connected to robotic agent and provides location data of the incidents to the
robot. Secondly, the middle-tier consists of the robotic agent which is the critical
component of the system that provides interaction, actuation, and mobility, as well as task
offloading. Finally, the top-tier cloud infrastructure provides the robotic network with a
computational platform to support the robotic agent offload tasks to the cloud and
complete them with more ease and in a timely manner.

However, due to the requirements and types of different applications, not all tasks are
possible or desirable to be offloaded. While a robot must perform some tasks (i.e.,
movement, interaction), some others may be offloaded if beneficial. Therefore, a proper
strategy (algorithm) is required to optimally allocate tasks to appropriate resources for
successful task completion under given constraints. In the following section, we will
formulate an optimization problem to identify the optimal task allocation/offloading
decisions for the given set of tasks that will result in the improvement of performance.
3.4.1 Mathematical formulation

We propose a framework for a crowd control system (Fig. 3.2) that utilizes a unified network of robotic and cloud infrastructure. Consequently, a proper task allocation between the robot and cloud is intended to make sure that performance enhancement is achieved. In this case, allocating the task to the cloud is referred to as task offloading. So, both are interchangeable for the context of this work. We use the application taskflow in Fig. 3.4 to define the optimization problem. Based on this, we then design a genetic algorithm-based offloading method to solve the problem. Using the GA-based algorithm, we identify the correct tasks to offload to the cloud, which results in an improvement of system outcome (i.e., lower energy, faster completion, more offloaded tasks etc.).

Table 3.1: Notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{total}$</td>
<td>Total robotic energy consumption for execution of the taskflow</td>
</tr>
<tr>
<td>$T_{total}$</td>
<td>Total execution time of the Taskflow</td>
</tr>
<tr>
<td>$T_{Deadline}$</td>
<td>Total allocated time for the execution of the taskflow</td>
</tr>
<tr>
<td>$E_{Limit}$</td>
<td>Energy consumption limit for the execution of the Taskflow</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Amount of bandwidth (fixed)</td>
</tr>
<tr>
<td>$N(t_i)$</td>
<td>Number of instructions for task $t_i$</td>
</tr>
<tr>
<td>$d_u(t_i)$</td>
<td>Amount of uploaded data for task $t_i$</td>
</tr>
<tr>
<td>$d_d(t_i)$</td>
<td>Amount of collected data for task $t_i$</td>
</tr>
<tr>
<td>$BPI$</td>
<td>Bits per instruction</td>
</tr>
<tr>
<td>$CPI$</td>
<td>Average number of clock cycles per instruction</td>
</tr>
<tr>
<td>$S_r$</td>
<td>Clock speed of robot</td>
</tr>
<tr>
<td>$S_c$</td>
<td>Clock speed of VM on cloud</td>
</tr>
</tbody>
</table>
We derive a task graph from the motivating application scenario (Fig. 3.4) [183], where a graph is presented as a sequence of dependent tasks to be completed by a robot under constraint. We model our application as a DAG to indicate our taskflow where each task is represented by a node. We present the DAG by using a tuple, $D = (\mathcal{T}, K)$. In this case, $\mathcal{T} = \{T_i, j = 1: t\}$ and $t = |\mathcal{T}|$. Here $\mathcal{T}$ denotes a task node. We also assume, $K=\{k_i, j = \langle t, t_j \rangle\}$ and $k = |K|$. $K$ implies a set of edges that refers to the communication cost from node $t_i$ to $t_j$. The term $t_i$ denotes a task $i$ in the task graph where its execution time is dependent on the computation of $t_i^{th}$ task with input data $d_i$. All the task nodes are indicated by Tasks $t_1$ ...... $t_T$. We also assume that the nodes on the same level of the DAG (e.g., Tasks 4 and 5) are independent and limited by the “dependency of precedence”. As a result, a task can start only after all its preceding tasks on the previous level have been completed. The highlighted nodes indicate tasks such as movement (task 19 in Fig. 3.4) and interaction (task 9 in Fig. 3.4) that are conferred upon the robot. Table 3.1 provides the necessary notation for the calculation of the cost functions. In designing our offloading approach, the goal is to find the optimal offloading decisions in order to complete the taskflow within the provided constraints. The following factors are taken into consideration for offloading decision-making: the processing capabilities of the robot and cloud VMs, movement cost, cost of robot-cloud communication and WSN communication [182].

i) Cost Functions for Energy Calculation

\begin{align*}
    E_{\text{total}} &= \sum_{i=1}^{[\mathcal{T}]} I_{t_i}. E_R(t_i) + \sum_{i=1}^{[\mathcal{T}]} \neg I_{t_i}. E_C(t_i) \quad (3.1) \\
    E_R(t_i) &= E_{\text{Mov}}(t_i) + E_{\text{WSN}}(t_i) + E_{\text{RC}}(t_i) \quad (3.2) \\
    E_C(t_i) &= E_{\text{Mov}}(t_i) + E_{\text{WSN}}(t_i) + E_U(t_i) + E_I(t_i) + E_{\text{idle}}(t_i) \quad (3.3)
\end{align*}
In this equation, $E_{\text{total}}$ is the total energy consumption of the robot, who is the centralized decision-maker for each task allocation. $I_t$ denotes the offloading decisions and $\neg$ stands for the NOT operator. The robotic energy costs of a task taking place on the robot and the cloud are represented by $E_R$ and $E_C$ respectively. We use $E_i$ and $E_U$ to indicate communication costs for sending task-related instructions and uploading collected data to the cloud. All these calculations are made based on the assumption that the robot doesn’t move significantly from a given access point while the communication is taking place (due to the type of application considered)

$$E_i(t_i) = P_i \times BPI \times \frac{N(t_i)}{\beta} \quad (3.4)$$

$$E_U(t_i) = \sum_{t_i \in v(t)} P_u \times \frac{d_u(t_i)}{\beta} \quad (3.5)$$

In equation (3.5), $v(t)$ is a set that characterizes tasks where robot collects data from sensors in WSN. $P_i$ and $P_u$ are the robot processing power for their corresponding communications.

$$E_{RC}(t_i) = P_r \times CPI \times \frac{N(t_i)}{S_r} \quad (3.6)$$

$$E_{idle}(t_i) = P_{idle} \times CPI \times \frac{N(t_i)}{S_c} \quad (3.7)$$

$$E_{WSN}(t_i) = \sum_{t_i \in D(t)} P_d \times \frac{d_d(t_i)}{r_r} \quad (3.8)$$

$$E_{Mov}(t_i) = \sum_{t_i \in M(t)} P_{mov} \times \frac{\sqrt{(x_a-x_b)^2+(y_a-y_b)^2}}{R_v} \quad (3.9)$$

In equation (3.6), $E_{RC}(t_i)$ indicates the energy of robotic computation energy, whereas $E_{idle}(t_i)$ in equation (3.7) defines the computation energy for a robot when a task is being executed on the cloud. $P_r$ and $P_{idle}$ indicate the processing power that the robot consumes during the respective computation processes. Also, $E_{WSN}(t_i)$ is the cost for connection with devices in the WSN. We consider $D(t)$ as the set of tasks for which the robot needs
to collect data. $T_r$ is the data transfer rate between the robot and WSN. The processing power of the robotic agent for data collection is indicated by $P_d$.

For movement, each zone is represented by its individual coordinates. We show three coordinates to imply that different events may take place in separate locations. As the robot moves from one zone to another, we calculate the shortest distance between two coordinates as the distance between the corresponding locations. For example, tasks that need movement from one zone $(x_a, y_a)$ to another $(x_b, y_b)$ are considered as part of the set $m(t)$. Based on this, we calculate the robot movement energy cost $E_{Mov}(t_i)$ in equation (3.9). As the robot may need movement for data collection as part of some tasks taking place in both the robot and the cloud, we consider the parameter $E_{Mov}(t_i)$ in equations (3.2) and (3.3). The robot velocity and power consumption during movement are $R_v$ and $P_{mov}$ respectively. We also assume the communication bandwidth $\beta$ of all zones to be different. So, the available bandwidth is a determining factor for offloading.

One major point to notice here is that the movement decisions are fixed with respect to the task. So, the robot will only go to fixed locations, which also constrains the distance covered by the robot. Moreover, for offloaded tasks, the bandwidths at the given locations are also considered as fixed values. This means that communication energy for the robot and the cloud only depends on task complexity (instruction size, data size etc.). In this situation, the simulation results will provide the optimal offloading decisions for a scenario with fixed movement and communication aspects. Later on, we will change these parameters to analyse their impact on offloading decisions and system performance.

\textit{ii) Cost Functions for Time Calculation}

\begin{equation}
T_{total} = \sum_{i=1}^{[T]} I_{t_i} \cdot T_R(t_i) + \sum_{i=1}^{[T]} -I_{t_i} \cdot T_C(t_i) \tag{3.10}
\end{equation}
\[ T_R(t_i) = T_{Mov}(t_i) + T_{WSN}(t_i) + T_{RC}(t_i) \] (3.11)

\[ T_C(t_i) = T_{Mov}(t_i) + T_{WSN}(t_i) + T_U(t_i) + T_{CC}(t_i) + T_I(t_i) \] (3.12)

The total task execution time is specified by \( T_{total} \). The term \( T_R \) indicates the time for tasks that are commencing on the robot and \( T_C \) means the execution time for tasks in the cloud VM. \( T_U \) and \( T_I \) are the communication costs for sending task-related instructions and uploading collected data to the cloud.

\[ T_I(t_i) = BPI \times \frac{N(t_i)}{\beta} \] (3.13)

\[ T_U(t_i) = \sum_{t_i \in v(t)} \frac{d_u(t_i)}{\beta} \] (3.14)

\[ T_{WSN}(t_i) = \sum_{t_i \in D(t)} \frac{d_d(t_i)}{T_r} \] (3.15)

\[ T_{MOV}(t_i) = \sum_{t_i \in M(t)} \sqrt{(x_a-x_b)^2+(y_a-y_b)^2} \] (3.16)

\( T_{Mov}(t_i) \) and \( T_{WSN}(t_i) \) are the respective values of the tasks that need movement and WSN communication. The computation time for tasks that are commencing on the robot and the cloud are \( T_{RC} \) and \( T_{CC} \) respectively.

\[ T_{RC}(t_i) = \frac{N(t_i) \times CPI}{S_r} \] (3.17)

\[ T_{CC}(t_i) = \frac{N(t_i) \times CPI}{S_c} \] (3.18)

The clock speed of the VM processor in cloud is considered to be \( M \) times faster than the robot (\( S_c = M \times S_r \)).

iii) Optimization Problem

With regards to the problem formulation and the presented cost functions, the objective is to find optimal offloading decisions for this application. We consider two scenarios for
simulation. In each case, the binary variable \( I_{t_i} = \{0, 1\} \) indicates the offloading decision options for a given task.

\( I_{t_i}(1) \) specifies that the task \( t_i \) is executed on the **robot**.

\( \neg I_{t_i}(0) \) specifies that the task \( t_i \) is executed on the **cloud**.

- **Scenario 1 (Minimise Energy):**

  In this proposed scenario, we have to obtain the optimal offloading decisions \( (I_{t_i}) \) where the objective is to minimise the robotic energy consumption (Minimize: \( E_{\text{total}} \)) and the constraint is task completion time/delay deadline \( (T_{\text{deadline}}) [182] \). That means,

  Find: \{ \( I_{t_i}\) \} for \( T = \{v_j, j = 1:t\} \) and \( t = |T| \) to

  \[
  \text{Minimise: } E_{\text{total}} \\
  \text{s.t.: } T_{\text{total}} \leq T_{\text{deadline}}
  \]

- **Scenario 2 (Minimise Time)**

  The objective is vice versa [183]. We have to find optimal offloading decisions \( (I_{t_i}) \) to minimise task completion time/delay (Minimise: \( T_{\text{total}} \)) under the energy constraint. That means,

  Find: \{ \( I_{t_i}\) \} for \( T = \{v_j, j = 1:t\} \) and \( t = |T| \) to

  \[
  \text{Minimise: } T_{\text{total}} \\
  \text{s.t.: } E_{\text{total}} \leq E_{\text{limit}}
  \]

  Based on this, we design a GA-based offloading scheme to solve the optimization problem for the abovementioned single taskflow of 20-node application.
3.4.2 Genetic algorithm-based scheme

As previously stated, the genetic algorithm (GA) is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. It represents an intelligent exploitation of random searches used in order to solve optimization problems. In GA, weak and unfit species are faced with extinction by the process of natural selection, whereas the strong ones have a higher possibility of passing their genes to future generations via reproduction [67]. GA is used to obtain optimized solutions from a number of candidate solutions [68]. Although randomized, GAs are by no means random; instead they exploit historical information to direct the search into the region of better performance within the search space. We use a genetic algorithm because it is a widely recognized global optimization algorithm that is used in many fields because of its high efficiency and impressive stability [65] [146]. In this thesis, we propose a GA-based task offloading scheme. The purpose of designing a GA-based method is to find out the optimal task offloading decisions (as mentioned in scenario 1 and 2) with respect to the given problem. It must be noted that we are only interested in GA to identify the solution as well as to verify the impact of a cloud-based approach in this application.
Figure 3.6: Encoding chromosome (single-layer) for task offloading decision-making space. As we are not concerned with the performance of GA in the optimization problems, we will not look into several other methods for obtaining the optimization solution. For the genetic algorithm-based offloading scheme, we need to follow the following regulatory steps [183] (as shown in Fig. 3.5).

1) **Chromosome encoding**: As seen in Fig. 3.5, the first step of designing the GA scheme is to encode the chromosomes (possible solutions) with respect to the problem set. In GA, a chromosome generally represents a unique solution (task offloading decision) for a problem. In our case study, we consider that an integer vector \( I = [I_{t_1}, I_{t_2}, \ldots, I_{t_i}, \ldots, I_{t_T}] \) corresponds to a solution where \( T \) is the total number of task nodes and each \( I_{t_i} \) contains the value either 0 (on cloud) or 1 (on robot). For example (Fig. 3.6), a chromosome \( I = [1 \ 0 \ 0 \ 1] \) indicates a solution where tasks 1 and 4 take place on a robot, whereas tasks 2 and 3 would take place on the cloud VM. Once a random chromosome is encoded, we modify the chromosome and constrain certain tasks (e.g., movement, interaction, etc.) of the chromosome by forcefully allocating them on the robot (as seen in Fig. 3.6). This is done to accommodate the tasks that can only be done by a robot because of its nature (highlighted in the task graph in Fig. 3.4).
ii) **Fitness evaluation**: The fitness function is a parameter that defines the quality of the proposed solutions in the search space of a generated population. Once the chromosomes/individuals in the population have been generated, they are evaluated, and fitness scores are acquired for each solution (Fig. 3.7). For the given problem, we have two scenarios and therefore two fitness measures.

Scenario 1 (Minimise: Energy), we consider the fitness measure, \( f = E_{\text{total}} \).

Scenario 2 (Minimise: Time), we consider the fitness measure, \( f = T_{\text{total}} \).

In both cases, the objective is to find the solution/chromosome that provides a solution with the lowest fitness measure. The lower fitness results in a better (more optimal) solution. In each generation, the total robotic energy consumption, task completion delay, and all other associated values are calculated for a given population using equations (3.1-3.18). The pseudo-codes for calculating total robotic energy consumption \( E_{\text{total}} \) and the

```plaintext
Input: Total task (T), DAG, I(\ell_\ell' \rightarrow \ell_i)
Output: Total robotic energy consumption (E_{\text{total}})
1: Initialize \( E_{\text{total}} \)
2: /*Calculate total robotic energy consumption*/
3: for each task level \( L_j \in L_{\text{total}} \) do
4:   for each task \( t_i \in L_j \) do
5:     Find Task Allocation \( L_j(I_{\ell_i}) \in (0,1) \)
6:     if \( I_{\ell_i} \) do /*Task on robot*/
7:       \( E_R(t_i) = E_{\text{mov}}(t_i) + E_{\text{WSN}}(t_i) + E_{\text{RC}}(t_i) \)
8:     else \( \neg I_{\ell_i} \) do /*Task on cloud*/
9:       \( E_C(t_i) = E_{\text{mov}}(t_i) + E_{\text{WSN}}(t_i) + E_U(t_i) + E_f(t_i) + E_{\text{idle}}(t_i) \)
10: \( E_{\text{total}} = E_{\text{total}} + E_{\text{RC}}(t_i) \)
11: end if
12: end for
13: end for
14: Return \( E_{\text{total}} \)
```

Figure 3.7: Pseudo-code for calculating the total robotic energy consumption
delay ($T_{total}$) are presented in Fig. 3.7 and 3.8 respectively. In both cases, the calculation for task graph is done level-wise. This means a taskflow is searched to find all the node dependencies between each level. These are then used to divide the task nodes into a number of groups. Tasks on each level are considered as a “task group”. Using all this information, we calculate the values of $E_{total}$ and $T_{total}$ for both scenarios.

• Calculation of $E_{total}$

For every task level, the robot checks each task for its allocation (robot or cloud). Based on this, values of $E_R(t_i)$, $E_C(t_i)$ and $E_{total}$ are updated. Then it moves on to the next level to complete the same task and add to the value of $E_{total}$. This is repeated until tasks on each level have been considered and calculated. At that point, it collects the result of $E_{total}$, which is the final value of total robotic energy for that proposed solution.

| Input: Total task (T), DAG, I ($I_{l_{t_i}}$, $-I_{l_{t_i}}$) |
| Output: Total time consumption ($T_{total}$) |
| 1: Initialize $T_{total}$ |
| 2: /*Calculate total time consumption*/ |
| 3: for each task level $L_j \in L_{total}$ do |
| 4: for each task $t_i \in L_j$ do |
| 5: Find task allocation $L_j(l_{t_i}) \in (0,1)$ |
| 6: if $l_{t_i}$ do /*Task on robot*/ |
| 7: $T_R(t_i) = T_{Mov}(t_i) + T_{SN}(t_i) + T_{RC}(t_i)$ |
| 8: else $-l_{t_i}$ do /*Task on cloud*/ |
| 9: $T_C(t_i) = T_{Mov}(t_i) + T_{SN}(t_i) + T_{U}(t_i) + T_{T}(t_i) + T_{idle}(t_i)$ |
| 10: end if |
| 11: end for |
| 12: if $T_R(t_i) > T_C(t_i)$ do |
| 13: $T_{total} = T_{total} + T_R(t_i)$ |
| 14: else do |
| 15: $T_{total} = T_{total} + T_C(t_i)$ |
| 16: end for |
| 17: Return $T_{total}$ |

Figure 3.8: Pseudo-code for calculating the total task completion time/delay
• Calculation of $T_{\text{total}}$

In the case of the calculation for $T_{\text{total}}$, the process is similar but slightly more complex. As tasks on the same level may happen in parallel, the total time for all tasks is not additive. Due to that, the process is slightly modified. Time calculation is done for each task level and the allocation is checked for each task. As seen in Fig. 3.8, for parallel tasks (also known as task groups) on the same level, time/delay for the tasks taking place on the cloud and the robot are calculated and updated on $T_R(t_i)$ and $T_C(t_i)$, respectively. Then a comparison is made between the two to find which consumes more time. For each task level, a comparison is made to collect the higher value between $T_R(t_i)$ and $T_C(t_i)$. That value is considered as the time needed to complete tasks on that particular level. It is then added to $T_{\text{total}}$. This is done for all the levels in order to obtain the final value of total completion time ($T_{\text{total}}$).

During the fitness evaluation phase, both scenarios have their own objectives and constraints. For the first scenario (Minimise: $E_{\text{total}}$) the constraint is task completion time/delay ($T_{\text{total}} \leq T_{\text{deadline}}$). For the second scenario (Minimise: $T_{\text{total}}$), the constraint is robotic energy ($E_{\text{total}} \leq E_{\text{limit}}$). Based on this, the fitness score is evaluated for every solution that meets the corresponding constraint. Every time there is a fitness score lower than the previous one, the lowest fitness score is updated. This way, the solutions are being improved until they reach the minimal fitness score.

iii) Selection phase: The selection phase follows the fitness evaluation (Fig. 3.5). In this phase, a mating pool is generated in order to gather solutions that are considered to be “good”. Thus, the chromosome with fitness measure lower than the average fitness measure are selected (note that solutions with lower fitness measure means better solutions here as explained in “fitness calculation” section) for the mating pool.
iv) **Crossover phase:** The crossover operation swaps segments between pairs of good solutions with the intention to produce offspring that represent better solutions. This is done by randomly selecting two chromosomes from the mating pool and producing a new chromosome from them by crossover. The newly created chromosome is placed in a list and the process is repeated. The process stops when the list reaches the size of the initial population. Fig. 3.9 demonstrates how two selected chromosomes produce an “offspring” through crossover. Here we consider uniform crossover, which uses a fixed mixing ratio between two parents. It enables parent chromosomes to contribute at the gene level rather than at the segment level. In this stage, individual bits in the string are compared between two parents. The bits are swapped with a fixed probability. As for the pre-defined tasks (on robot), the crossover phase doesn’t hamper the results, as all possible chromosomes have already been modified during chromosome encoding to compensate for that.
v) **Mutation phase:** The mutation phase takes place on the list of chromosomes produced from the crossover phase. There is now a new population full of possible solutions (task offloading decisions). In some cases, the chromosomes may become too similar to each other. Therefore, some randomly selected loci in the gene value is altered with a certain probability (0.5 in our case) in order to have a higher chance of finding a global optimum (see Fig. 3.10). By doing so, GA not only accelerates the convergence to the optimal solution, but it also potentially prevents premature convergence by maintaining diversity. In addition, a further chromosome encoding is done at this stage to retain the previous fixed allocations in case one of the genes containing the pre-fixed allocations have been changed.
vi) Stopping criteria: The final phase of the GA operation is to define the stopping criteria. Once the new population (after the mutation phase) replaces the current population, we move on to the next generation to continue the same process. We input a stopping criterion in GA to find out when the optimal result is obtained. In our case, since GA will provide its “best fitness measure” after each generation, we stop when the best fitness measure does not change after a prefixed number of generations. At that point, the GA scheme stops running and results (offloading decisions) are considered optimum.

3.4.3 Simulation setup and offloading performance

In this section, we calculate and analyse the results of our given problem. The 20-node graph presented in Fig. 3.4 is used as our taskflow for two simulations. In the first simulation, the objective is to find the offloading decisions that minimise $E_{total}$ under the time constraint. Then, we show the adaptability of the algorithm by showing the second simulation for a case where the objective is to complete the tasks in minimum time (Minimize: $T_{total}$). For this scenario, the constraint is total robotic energy consumption.

We compare the offloading results of the GA with two benchmarks. One is the Exhaustive Search (ES). In ES, all the possible solutions are systematically enumerated to find the optimal one. The purpose here is to verify the accuracy of the GA algorithm, rather than the performance. Additionally, we compare the results with an “all on robot” (AoR) approach where all tasks are considered to be completed by a robot. This result is used to evaluate if the alternative approach with the usage of the cloud infrastructure is beneficial or not. Finally, we compare with a greedy algorithm method that considers the optimal/best solution (0 or 1) from each task node to calculate an overall result. These comparisons help validate our designed method with respect to the established benchmarks.
Table 3.2: Simulation parameter setup for the 20-node taskflow

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimize $E_{total}$</th>
<th>Minimize $T_{total}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint</td>
<td>Time (60 s)</td>
<td>Energy (4000 J)</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Zone 1</td>
<td>Zone 2</td>
</tr>
<tr>
<td></td>
<td>256 Kbps</td>
<td>512 Kbps</td>
</tr>
<tr>
<td></td>
<td>Zone 3</td>
<td>128 Kbps</td>
</tr>
<tr>
<td>Task Constraint</td>
<td>Tasks 5, 9, 15, 19, and 20 must be done by the robot</td>
<td></td>
</tr>
<tr>
<td>Stopping Criteria</td>
<td>300 Generations without change</td>
<td></td>
</tr>
</tbody>
</table>

i) Simulation Setup

For the simulation setup, we choose some typical values for system parameters. For instance, the CPU (Intel Core i5-4570) of the robot is equipped with processing clock speed of 3.2 GHz, CPI value of 20, and BPI value of 8. The VM clock speed (cloud) is 1000 times faster than the robot CPU. For a typical robot with an average CPU power rating of 84 W, the other processing powers are presented in the following way: $P_r = P_{idle} = 50$ W, $P_i = 35.5$ W, $P_u = P_m = 80$ W, and $P_d = 35$ W.

As seen in Fig. 3.4, Tasks 1, 2, and 14 require a robot to collect data from a nearby available sensor network. Tasks 5, 9, 15, 19, and 20 are constrained to be completed by the robot and hence the GA-based scheme is modified accordingly. In particular, movement-based tasks result in a change of location/zone and available bandwidth for the robot. The taskflow in Fig. 3.4 represents two movement-based tasks (task 5 and 9). This is why three zones have been considered for the simulations. Table 3.2 presents all the necessary setup details of parameters. For both case, the simulation is run with a defined stopping criterion (Table 3.2). The optimal results from simulations are collected to analyse and compare the performance of the scheme in comparison to other methods.
Figure 3.11: Performance graph of 20-node taskflow (Minimise: $E_{total}$)

Table 3.3: Performance comparison for 20-node taskflow (Minimise: $E_{total}$)

<table>
<thead>
<tr>
<th>Result Parameters</th>
<th>Genetic Algorithm</th>
<th>Exhaustive Search</th>
<th>All on Robot</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation No</td>
<td>371</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Offloading Decisions (1-Robot) (0-Cloud)</td>
<td>1 1 0 0 1 0 1</td>
<td>1 1 0 0 1 0 1 0</td>
<td>1 1 1 1 1 1 1</td>
<td>1 1 0 0 1 0 0</td>
</tr>
<tr>
<td>Offloading Decisions (1-Robot) (0-Cloud)</td>
<td>0 1 1 1 0 1 1</td>
<td>1 0 1 0 1 1 0 1 1</td>
<td>0 1 1 1 1 1 1</td>
<td>0 1 0 1 0 1 1</td>
</tr>
<tr>
<td>Offloaded Task</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Minimum Energy ($E_{total}$)</td>
<td>2746.98 J</td>
<td>2699.13 J</td>
<td>3802.23 J</td>
<td>8632.44 J</td>
</tr>
<tr>
<td>Completion Time</td>
<td>52.40 s</td>
<td>51.74 s</td>
<td>70.01 s</td>
<td>101.32 s</td>
</tr>
<tr>
<td>GA Overhead Time</td>
<td>3.96 s</td>
<td>411.72 s</td>
<td>N/A</td>
<td>2.11 s</td>
</tr>
<tr>
<td>GA Overhead Energy</td>
<td>332.36 J</td>
<td>34586.11 J</td>
<td>N/A</td>
<td>179.35 J</td>
</tr>
<tr>
<td>Overall Time= Completion Time + GA Overhead Time</td>
<td>56.36 s</td>
<td>463.46 s</td>
<td>70.01 s</td>
<td>93.43</td>
</tr>
<tr>
<td>Overall Energy = Completion Energy + GA Overhead Energy</td>
<td>3079.74 J</td>
<td>37285.24 J</td>
<td>3802.23 J</td>
<td>8811.79 J</td>
</tr>
</tbody>
</table>
ii) Simulation Results

We run a simulation of a GA-based scheme for a single 20-node taskflow to find the optimal offloading decisions. To show the adaptability of GA in this context, we considered two scenarios. In case there is any shortage of one of the parameters (energy/time), the robotic agent can prioritize between the two and adjust the task objectives accordingly.

- Scenario 1 (Minimize: $E_{\text{total}}$)

From the simulations with fitness measure $E_{\text{total}}$, the results show that the average fitness measure decreases with the generation number (Fig. 3.11). This leads to the conclusion that most generations of the GA scheme tend to result in a lower fitness measure and thereby decrease the average fitness. The lower trend of the graph clearly indicates a process where the GA keeps replacing the fitness with lower scores (which in this case means better energy measure) until it finds the minimal one. From generation 371 onwards, no change in generation score is seen. Hence, the GA stops running after 371 generations in accordance with the stopping criteria (Table 3.2).

The results of the simulation are presented in Table 3.3, along with two benchmarks for comparison. As seen from the table, an offloading decision for each task is provided. According to the results, 13 tasks take place on-board the robot (represented by “1”), which means seven tasks have been offloaded (represented by “0”). As the completed task allocation result is presented in a series, the seven offloaded tasks (Tasks 3, 4, 6, 8, 12, 16, and 18) is clearly identified. This is the optimal offloading decision for each task of the single taskflow.

A further comparison with the ES helps to verify the accuracy of the algorithm. We observe that ES presents the minimal energy (2699.13 J), which is slightly better than the
GA result. The situation is the same for the task completion time (51.74 sec), which means that both algorithms manage to meet the delay constraint of 60 sec. A more in-depth analysis shows that the reason for the difference in the value of minimal energy is the number of offloading decisions. The number of tasks offloaded for ES (8) is greater than GA (7). As a result, more offloading in ES provides the better value. However, GA provides the near-optimal result. Even though the results for minimal energy in ES are slightly better, it has some drawbacks. The major difference between the two methods is in the overhead. ES takes about 511.74 sec to run the algorithm, which is more than 100 times the overhead time needed to run the GA (3.96 sec). At the same time, the algorithm overhead energy for GA is 332.36 J. In the case of ES, the overhead energy is 34586.11 J, which is significantly higher. As the robot has limited energy, ES is definitely not a suitable option as it costs significantly more overhead time and more energy. Moreover, as the scale of the operation gets larger, computation will increase more, at which point the ES overhead will be too high to justify the optimal solution it provides. From this point of view, the GA performs better for the proposed scenario as it provides a near-optimal result with a slight error in the overall minimum energy, but in significantly less overhead time.

We conduct another comparison for GA with an AoR (all-on-robot) approach. The results in Table 3.3 show that the processing energy (3802.23 J) of AoR is very high when compared with GA. More significantly, the task completion time in this case is 70.01 sec, which does not meet the delay constraint. As AoR does not have any scope of adaptability, it is definitely not a suitable solution in this context. In contrast, the offloading approach to the cloud with GA provides more opportunities for the robot to adapt and complete the necessary tasks in cases when the AoR does not work.
Finally, the greedy algorithm considers the optimal solution for each task node to obtain the simulation results. But this is not always the best approach. In cases of parallel tasks, it may be a better option to allocate tasks to separate resources (0 or 1) than to choose the same resource (as calculated by greedy) which might be unavailable and result in additional idle time/energy. Thus, total energy for greedy (8632.44 J) is high and total time (101.32 sec) is outside the constraint, even though more tasks (10) are offloaded than GA. So, it’s not suitable in this case.

Finally, we have evaluated the efficiency of the GA by calculating the “overall time”, which includes the GA overhead time as well as the task completion time. It is found that the overall time (56.36 sec) is less than the time constraint of 60 sec. So, it is observed that in terms of overall energy, the GA-based scheme meets the delay constraint and is lower than the other benchmarks (463.46 sec for EX, 70.01 sec for AoR and 101.32 sec for greedy), as seen in Table 3.3. Even for the overall energy, the value for GA-based scheme is 3079.74 J, which is lower than ES (37285.24 J), AoR (3802.23 J) as well as the greedy algorithm (8811.79 J).

The results from this section is used a benchmark for later simulations, where solutions are only considered as feasible, when it meets the time/energy deadline even after the inclusion of GA overhead. Furthermore, simulations in the following chapters will also be conducted on more complex DAGs to see the performance of its accuracy in a more complex scenario, as well as observe its overhead to check if the overall time/energy is below deadline. It will also give us a chance to identify if the added layers of complexity in decision-making for our GA method leads to increased overhead as well.
Figure 3.12: Performance graph of 20-node taskflow (Minimise: $T_{total}$)

Table 3.4: Performance comparison for 20-node taskflow (Minimise: $T_{total}$)

<table>
<thead>
<tr>
<th>Result Parameters</th>
<th>Genetic Algorithm</th>
<th>Exhaustive Search</th>
<th>All on Robot</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation No</td>
<td>651</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Offloading Decision (1-Robot)</td>
<td>1 1 0 0 1 0 1</td>
<td>1 1 0 0 1 0 1</td>
<td>1 1 1 1 1 1 1</td>
<td>1 1 0 1 1 0 1</td>
</tr>
<tr>
<td>(0-Cloud)</td>
<td>0 1 1 1 0 1 1</td>
<td>1 1 0 1 1 0 0 0</td>
<td>1 1 1 1 1 1 1</td>
<td>0 1 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0 0 1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 0 0 0 1 1</td>
</tr>
<tr>
<td>Offloaded Task</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Minimal Time ($T_{total}$)</td>
<td>50.12 s</td>
<td>50.12 s</td>
<td>70.01 s</td>
<td>89.12 s</td>
</tr>
<tr>
<td>Total Energy</td>
<td>2766.95 J</td>
<td>2766.95 J</td>
<td>3802.23 J</td>
<td>11214.54 J</td>
</tr>
<tr>
<td>Overhead Time</td>
<td>10.70 s</td>
<td>661.34 s</td>
<td>N/A</td>
<td>3.22 s</td>
</tr>
<tr>
<td>Overhead Energy</td>
<td>838.53 J</td>
<td>55552.28 J</td>
<td>N/A</td>
<td>273.7 J</td>
</tr>
<tr>
<td>Overall Energy = Completion Energy + GA Overhead Energy</td>
<td>3605.48 J</td>
<td>58319.23 J</td>
<td>3802.23 J</td>
<td>11488.15 J</td>
</tr>
<tr>
<td>Overall Time = Completion Time + GA Overhead Time</td>
<td>60.82 s</td>
<td>711.46 s</td>
<td>70.01 s</td>
<td>92.34 s</td>
</tr>
</tbody>
</table>
Scenario 2 (Minimize: $T_{total}$)

We run simulations for an alternative scenario where the objective is to minimise the time. Similar to the previous simulation, the downward trend of average fitness measure (Fig. 3.12) suggests that the GA is working properly. The lowest fitness measure looks for even lower values to be replaced by, until finding the lowest one. For this scenario, it takes 651 simulations to find the minimum task completion time, which is a lot higher than the previous scenario. The optimal offloading solution (Table 3.4) shows the eight tasks being offloaded to the cloud (i.e., tasks 3, 4, 6, 8, 12, 16, 17, 18, which are represented by “0”).

We also compare with ES, AoR and greedy to verify the performance. For ES, we get the same result as GA, which means it is an optimal result. As ES has a high overhead energy and time (55552.2836 J and 661.34 sec, respectively), it is not suitable. However, for GA, the overhead energy (838.53 J) and time (55552.28 J) is around 60 times less. The results of AoR show that it takes 70.01 sec to complete the taskflow. Even though the total energy consumption (3802.23 J) meets the energy constraint, the method does not result in the minimum task completion time. Instead, a GA-based approach provides the best result (as shown earlier). Finally, minimal time for greedy is 89.12 sec, whereas energy is 11214.54 J, which is outside the limitation (4000 J). So, the greedy algorithm is a not suitable option.

Furthermore, we have added an extra criterion to accommodate the GA overhead energy and time to the overall energy and time calculations. The objective here is to see if the added GA overhead can still maintain the performance within constraints. As seen in Table 3.4, the GA-based scheme provides overall energy of 3605.48 J, which is less than the energy constraints. It is also lower than the other benchmarks of ES (58319.23 J), AoR (3802.23 J) and greedy (11488.15 J). The same is found in case of overall time.
ES, AoR and greedy take 711.46 sec, 70.01 sec and 92.34 sec respectively, while the value for the GA-based scheme is 60.82 sec. The results are a clear indication that even with GA overhead, the algorithm still manages to find the optimal results and outperforms the other methods in terms of objective. The simulations in following chapters has also followed the same principle.

All the performance criteria (e.g., time, energy, overhead, etc.) in both scenarios point to the superiority of the genetic algorithm (GA)-based scheme over the other three benchmarks of Exhaustive Search (ES), All on Robot (AoR) method and greedy algorithm.

3.4.4 Impact of communication and mobility on offloading

In the following section, we analyse the impact of communication and the mobility aspect by varying key parameters (zone distance and bandwidth) to see the changes in system outcome due to these factors.

- Simulation by varying bandwidth (minimize: $T_{total}$)

We run simulations to verify bandwidth to see its impact on offloading decisions and system performance. The results are presented in Table 3.5. We considered scenario 2 (Minimize: $T_{total}$) for running the simulations of bandwidth change. As mentioned, we verified the adaptability of the GA-based scheme with respect to bandwidth. As different parts of the above-mentioned application take place in different locations, we varied the bandwidth values for one of the zones/locations (Zone 2) to observe the effects of bandwidth change. We ran five different simulations and evaluated the results of the offloading decisions and minimum completion time.
Table 3.5: Impact of bandwidth change on task offloading

<table>
<thead>
<tr>
<th>Bandwidth of Zone 2 (Mbps)</th>
<th>Minimum Time (Sec)</th>
<th>Total Robotic Energy (J)</th>
<th>Task Allocation [1-Robot][0-Cloud]</th>
<th>Offloaded Task No</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.128</td>
<td>51.15</td>
<td>2870.41</td>
<td>1 1 0 0 1 0 1 0 1 1 1 0 1 1 1</td>
<td>7</td>
</tr>
<tr>
<td>0.512</td>
<td>50.12</td>
<td>2766.95</td>
<td>1 1 0 0 1 0 1 0 1 1 1 0 1 1 1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>49.54</td>
<td>2720.21</td>
<td>1 1 0 0 1 0 1 0 1 1 0 0 1 1 1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>48.73</td>
<td>3519.53</td>
<td>1 1 0 0 1 0 1 0 1 1 0 0 1 1 1</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>48.61</td>
<td>2985.04</td>
<td>1 1 0 0 1 0 1 0 1 1 0 0 1 1 1</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>43.39</td>
<td>2689.87</td>
<td>1 1 0 0 1 0 0 0 1 0 0 0 1 1 1</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 3.13: Impact of bandwidth change on minimum completion time (min: \( T_{total} \)
Figure 3.14: Impact of bandwidth change on offloading decisions (min: $T_{total}$)

The results suggest a clear decline in task completion time with the increase in bandwidth (illustrated in Fig. 3.13). At the same time, the bandwidth increase also ensures a clear progression for the number of offloading decisions (Fig. 3.14). This indicates that better bandwidth enables the GA scheme to adapt to changing conditions and offload more tasks, which in turn decreases the minimum task completion time of the robot. It also means that bandwidth has a significant influence on the offloading decisions and thereby how well the system performs. Similar results have been found for scenario 1 (Minimize: $E_{total}$), where the GA adapts with the changing bandwidth and provides more offloading options for tasks. Thus, it manages to find the optimal offloading decisions that provide minimal energy while meeting the constraints.

This suggests that changing the bandwidth has a significant impact on offloading for robotic operations. As the robot has the ability to move, this opens up the area for discussion regarding the impact of bandwidth and movement on performance as well as each other. A further study is required to analyse its impact and find ways to utilize it properly for improving system outcome.
• Simulation by varying movement/distance (minimise: $E_{\text{total}}$)

To understand the impact of movement on the current scenario, we have selected scenario 1 ($\text{Minimise: } E_{\text{total}}$) to run simulations. The three zones in our problem set are represented by coordinates. We have changed the coordinates of one of the zones (zone 2) to increase the distance between zone 1 and zone 2. For that changing condition, we run simulations on two cases. In the first case, the bandwidth of zone 2 is 512 Kbps. This section explains the results for this particular case.

From the results in Table 3.6, we see that there is an increasing trend for both energy and time. This means that the increase in distance between zones results in more robotic energy consumption as the movement itself is a task that requires energy. This also increases the total completion time. As seen from the table, the total completion time becomes higher than the time constraint (60 sec) after a certain amount of increase in distance. At that point, the robot fails to complete the tasks within the given constraints despite offloading seven tasks. For this simulation, this happens when the zone 2 coordinates result in a distance of 54.43 units between zone 1 and zone 2. These results clearly show that movement has a significant impact on task offloading decisions.

• Joint effect of movement and communication

We further study the integrated impact of both bandwidth and movement in task offloading decisions [183]. This also helps us to show how this may be used to our benefit. For the simulation results in Table 3.6, the first case (512 Kbps) presents a situation where the increasing distance causes the movement energy and time to become so high that the total completion time exceeds the constraints. Even though seven tasks were offloaded using available bandwidth (512 Kbps), it was still not enough.
### Table 3.6: Impact of movement on offloading decisions and system performance

<table>
<thead>
<tr>
<th>Zone Locations (x, y)</th>
<th>Distance Between Zone 1 and Zone 2 (Units)</th>
<th>Case 1: Zone 2 BW (512 kbps)</th>
<th>Case 2: Zone 2 BW (5 Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Minimal Energy (J)</td>
<td>Total Time (Sec)</td>
</tr>
<tr>
<td>Zone 1: (7.70,9.31)</td>
<td>6.08</td>
<td>2746.98</td>
<td>50.39</td>
</tr>
<tr>
<td>Zone 2: (2.08,11.65)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zone 1: (7.70,9.31)</td>
<td>14.56</td>
<td>2882.74</td>
<td>52.09</td>
</tr>
<tr>
<td>Zone 2: (22.08,11.65)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zone 1: (7.70,9.31)</td>
<td>34.46</td>
<td>3200.99</td>
<td>56.07</td>
</tr>
<tr>
<td>Zone 2: (42.08,11.65)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zone 1: (7.70,9.31)</td>
<td>54.43</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Zone 2: (62.08,11.65)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To solve this situation, we ran simulations for a second case. In this case, everything remained the same except for the bandwidth of one of the zones (zone 2). We increased the zone bandwidth to 5 Mbps. From the results, we see that the performance improved much faster. For the same increase in distance, the robot managed to complete the taskflow for all the simulations. Even when the distance increased to 54.43 units, it was still within the time constraint. This was possible because of better network connectivity (communication aspect), which meant faster offloading was possible. Also, the robot managed to offload nine tasks (more than in the other case), which also contributed to the improvement in the system results.
From the results and analysis, it is easy to conclude that movement and bandwidth make a significant impact on task offloading decisions as seen in our simulation for a single taskflow application of robotic crowd control. In fact, as the latter case suggested, a proper trade-off between movement and bandwidth has the capacity to actually overcome the shortcomings of the system and improve the performance when necessary. In addition, pre-gathered knowledge about the bandwidth and the locations could be useful for the robot to plan paths for offloading and task completions. We will further focus on this in the following chapters.

3.5 Multi-Taskflow Path Planning for Optimal Offloading

In addition to the task offloading problem in the previous section, we have introduced a “travelling salesman problem” for multi-taskflow optimal task sequencing in the given cloud networked robotic system. This section will further highlight the concept of offloading-oriented movement and communication link choice in the context of CNR.

Figure 3.15: Optimal task sequence for a multi-taskflow offloading problem
3.5.1 Problem setup

For this problem set, the robot is called upon to complete multiple taskflows at the same time. In this scenario, each taskflow needs to be completed within a given constraint (time) and with minimum energy (for each taskflow) by finding optimal offloading decisions. These different taskflows indicate the different locations for robot’s movement. So, it requires an optimal ordering of the taskflows to make sure that all the tasks sets are completed successfully with minimum energy. As one of the main variable factors in this problem is movement energy, the robot’s task ordering and resulting movement (path plan) need to be optimal.

The objective is to complete all the taskflows within the individual time constraints (for each of the taskflows) while expending the individual minimum energy requirement. Additionally, the optimal task sequence needs to be found in order to make sure that the cumulative energy for the taskflows is minimal.

3.5.2 Mathematical implementation

For our scenario, we assume that the robot is asked to complete $K$ number of taskflows at a given moment. Each taskflow is represented by $W$. For taskflow, $W = 1...K$, robot needs to find the optimal task ordering sequence resulting in movement to corresponding zones. Here the term $C_{E}$ signifies the cumulative Energy for all the taskflows. So,

$$C_{E} = \sum_{W=1}^{K} W_{E_{total}}$$  \hspace{1cm} (3.19)

In this equation, the term $W_{E_{total}}$ points to the total energy for each of the given taskflows. As mentioned before, for each of the taskflows the robot needs to find the optimal offloading decisions to meet the time constraint and find the minimum energy. In the previous section, we presented our GA-based offloading scheme to solve the task
offloading problem. On top of this, we add the scenario where the robot needs to complete all these taskflows by moving to their given locations. As different taskflows are taking place in different start and finish locations, it compels the robot to plan its sequence for moving to these various locations in a specific order to expend the minimum cumulative energy. Let $O_w$ indicate optimal order of task sequence of the taskflows. So, the objective of this problem is,

Find: $\{ O_w \}$ for $\{w = 1:K\}$ to

$$\text{Minimize: } C_E$$

In this case, each of the taskflow solves the task offloading optimization problem according to the Genetic Algorithm-based scheme, which is mentioned in the previous section.

3.5.3 Workspace setup and methodology

As presented in the workspace in Fig. 3.15, each of the taskflows requires the robot to first move from the current location to a given location, which is known as the investigation phase. As the robot moves to the new location, the investigation ($I$) phase begins. This is followed by the final phase, where the robot initiates the guide ($G$) event to move to the final destination. So, for each taskflow the robot completes its tasks in three separate zones (depending on the taskflow locations). However, in some cases, the finishing location of one phase may be the starting location of the next phase. In this case, the movement and its corresponding energy are considered to be 0. The location information for each of the taskflow is presented in Fig. 3.15.
Table 3.7: Performance of GA-based offloading for each taskflow (min: $E_{total}$)

<table>
<thead>
<tr>
<th>Taskflow Set</th>
<th>Coordinates (x,y) [S-Starting Phase] [I-Investigative Phase] [G-Guide Event Phase]</th>
<th>Optimal Energy (J)</th>
<th>Total Time (Sec)</th>
<th>Task Allocation No.</th>
<th>Task Offload No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I=(10.21,72.4) [Zone 5]</td>
<td>5339.81</td>
<td>92.24</td>
<td>1 1 1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G=(51.12,41.2) [Zone 3]</td>
<td></td>
<td></td>
<td>0 1 1 0 0 0</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>I=(32.1,60.3) [Zone 7]</td>
<td>5243.22</td>
<td>90.88</td>
<td>1 1 1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G=(56.3,96.1) [Zone 8]</td>
<td></td>
<td></td>
<td>0 0 1 0 0 0</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>I=(56.3,96.1) [Zone 8]</td>
<td>8121.15</td>
<td>139.2</td>
<td>1 1 1 1 1 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G=(16.09,16.65) [Zone 2]</td>
<td></td>
<td></td>
<td>1 1 1 1 1 1</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>I=(30.42,3.11) [Zone 3]</td>
<td>4669.39</td>
<td>83.71</td>
<td>1 1 1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G=(30.42,3.11) [Zone 6]</td>
<td></td>
<td></td>
<td>0 0 1 1 0 0</td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>I=(10.21,72.4) [Zone 5]</td>
<td>5276.81</td>
<td>95.53</td>
<td>1 1 1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G=(71.12,21.2) [Zone 4]</td>
<td></td>
<td></td>
<td>0 0 1 1 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

Table 3.8: Simulation results for multi-taskflow optimal task sequence

Optimal Task Set Sequence: [Task 1--- Task 4--- Task 5--- Task 2--- Task 3]

Minimal Energy of Task Set Sequence: **28713.3923 J**

Optimal Time of Task Set Sequence: **498.3585 sec**

For this given workspace, the robot needs to find the optimal task sequence for movement. To solve this problem, we initially run a GA-based algorithm for each of the taskflows to find the optimal offloading decisions that meet the constraints (in each case) and require the minimum robotic energy consumption. In addition, we run an exhaustive
search (ES) method to check all the possible combinations of the given task sequence. During this stage, it is important to keep track of the movement of the robot, as the ending location of one of the taskflows is the automatic starting point for the next taskflow. We also calculate the cumulative energy ($C_E$) by adding all of the values and compare it with the other possible solutions to find the lowest one. At the end of the calculation, $O_w$ characterizes the optimal sequence of tasks that results in the lowest $C_E$.

### 3.5.4 Simulation setup and results

For the simulation we consider the number of taskflows, $K = 5$. Each of the taskflows has two locations ($I$ and $G$) for investigation and guide event phases. The starting location of the robot, $S = (1.71, 3.32)$. The movement energy and time between each location (represented by coordinates) is found from equations (3.9) and (3.16), respectively. The complete workspace is divided into six zones. Each zone has its dedicated bandwidth (or no Internet). The available bandwidth is a critical factor for the robot in that location to communicate with the cloud. The information regarding the available bandwidth at each location is presented in Fig. 3.15. In this case we consider low rates for bandwidth so that the error rate would be low. As for the locations of the taskflows, all of them are associated with different zones. Based on all this information, a simulation is run to find the optimal task sequence. This means that the robot finds the optimal order in which to move so as to minimise the cumulative energy.

The results from the simulation are depicted in Fig. 3.15. We also provide the results for optimal offloading decisions of each of the taskflows (that meet their respective constraint). The results in Table 3.7 present the performance of each of the taskflows with respect to their own constraints for optimization. As the GA-based algorithm was implemented (as in the previous section), the results provide the optimal offloading
decision. Depending on the location of the tasks, the number of offloaded tasks may vary. For example, taskflow 2 consists of an investigation phase in zone 7, which has a bandwidth of 1 Mbps. This is reflected in the offloading decisions, with nine tasks being offloaded. On the other hand, taskflow 3 starts in zone 6, which has no Internet connection. As a result, all the tasks are done on the robot, which impacts the outcome significantly (high value for optimal energy as well as the constraint). These suggest the bandwidth has a major influence.

In the second part (Table 3.8), we present the results for optimal task sequence for the multi-taskflow offloading problem. Fig. 3.15 presents the task sequence visually. It depicts the optimal path from the starting position to the end of the set of taskflows. From the exhaustive search, we get the optimal combination for task sequence. It suggests the desired order to be: (Task 1---Task 4---Task 5---Task 2---Task 3) as seen in Table 3.8 along with the minimal cumulative energy of the task set sequence. In this way, the robot plans the optimal path to move in order to complete the set of taskflows via task offloading. This presents a simple scenario where offloading, path planning and choice of bandwidth have been integrated with the taskflow for smart city applications.

3.6 Summary

In this chapter, we initially presented the framework of CNR for optimal offloading. We also explained the possible applications of our framework: smart manufacturing and smart city applications. This led to our first proposed application of robotic crowd control, which is presented as a DAG (a single taskflow). We formulated our optimization problem to find the optimal offloading decisions for a single taskflow where movement and communication aspects are fixed. To solve the problem, we then designed a GA-based offloading scheme. Here we presented two scenarios to show the adaptability of
our scenario. Thorough simulation was run to analyse the results that suggest that the GA-based scheme optimally identifies the offloading decision for each task. We also compared with benchmarks (exhaustive, all-on-robot and greedy) to verify the accuracy of the results. The comparison advises that the GA-scheme solutions are near-optimal but with less overhead. Also, they outperform the abovementioned validated approaches. Additionally, we studied the influence of mobility and communication aspects by separately varying bandwidth and zone distance in two scenarios. It showed their individual and joint influence on offloading decisions as well as system performance.

We further studied their relationship for a simulation of multi-taskflow application. Here the robot needed to find the right sequence to complete five sets of DAGs (crowd control application). Each set of the solution had GA-based optimal results, while the optimal sequence was found through exhaustive means. This result highlighted that the mobility of the robot is interconnected with the access to bandwidth in different zones that eventually influences task offloading decisions as well. This motivates our work in the following chapter where we try to analyse and utilize the interdependency among mobility (path planning), communication (AP selection) and task offloading in the context of CNR in order to gain enhanced system output.
Chapter 4.

Communication-Aware Optimal Task Offloading for Mobile Cloud-Assisted Robot

The relationship among mobility, communication and offloading is exploited in this chapter to prepare a task offloading mechanism for a mobile cloud-assisted robot in the context of an industrial application. This helps us prepare a mobility-driven and communication-aware offloading strategy for our use case—smart factory maintenance. We present the application taskflow, workspace design and communication model in order to formulate a multi-objective optimization problem. A GA-based scheme is then designed where path planning (mobility), AP selection (communication) and offloading decisions are all considered as variables. Through simulation results and comparison with previous techniques (GA with fixed mobility and communication), we find out that considering path planning and AP selection as decision variables results in better system outcome. This also leads to energy-distance weighted sum for fitness score, where the robot has the potential to control how much effort is put on movement in relation to energy and time. We also associate a recharge-based offloading strategy where the robot plans its path and offloading to accommodate recharging during an application. In summary, the inclusion of mobility and communication aspects in offloading decision-making can be utilized in beneficial ways, thus bringing about superior performance in CNR systems.

4.1 Introduction

The “on-demand mobility” allows a robot to move to selected locations and access suitable communication links for faster offloading between itself and the cloud. This inspires the concept of our mobility-driven and communication-aware task offloading in
Table 4.1: Related work for offloading, AP selection and path planning

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method/Implementation</th>
<th>Task Offloading</th>
<th>Path Planning</th>
<th>AP Selection</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>[105]</td>
<td>Polynomial-Time Heuristic</td>
<td>✓</td>
<td></td>
<td></td>
<td>Minimize Energy</td>
</tr>
<tr>
<td>[184]</td>
<td>Offloading Framework (Ternary Decision Maker)</td>
<td>✓</td>
<td></td>
<td></td>
<td>Minimize Energy &amp; Time</td>
</tr>
<tr>
<td>[185]</td>
<td>Modified Genetic Algorithm</td>
<td></td>
<td>✓</td>
<td></td>
<td>Minimize Distance</td>
</tr>
<tr>
<td>[186]</td>
<td>User Cooperative Moving</td>
<td></td>
<td>✓</td>
<td></td>
<td>Maximize Throughput</td>
</tr>
<tr>
<td>[34]</td>
<td>Fully Polynomial Time Problem Approximation Scheme (FPTAS)</td>
<td>✓</td>
<td></td>
<td></td>
<td>Minimize Time</td>
</tr>
<tr>
<td>[187]</td>
<td>Framework (Lightning)</td>
<td></td>
<td>✓</td>
<td></td>
<td>Minimize Time</td>
</tr>
<tr>
<td>[148]</td>
<td>Bacterial Foraging Optimization</td>
<td></td>
<td>✓</td>
<td></td>
<td>Shortest Path</td>
</tr>
<tr>
<td>[188]</td>
<td>Genetic Algorithm</td>
<td>✓</td>
<td></td>
<td></td>
<td>Minimize Time</td>
</tr>
<tr>
<td>[99]</td>
<td>Cloud-Based Robot Navigation Assistance</td>
<td></td>
<td>✓</td>
<td></td>
<td>Compare Offloading Models</td>
</tr>
<tr>
<td>[153]</td>
<td>No Regret Learning Algorithm</td>
<td></td>
<td></td>
<td>✓</td>
<td>Max User Throughput</td>
</tr>
<tr>
<td>[136]</td>
<td>Lightweight Heuristic Method</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Minimize Delay/Time</td>
</tr>
<tr>
<td>[Chapter 4]</td>
<td>GA based Mobility-Driven and Communication-Aware Offloading</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Minimize Energy/Time/Distance</td>
</tr>
</tbody>
</table>
cloud-assisted robotic applications. The key aspects of communication and mobility we consider in this work are: AP selection and path planning, both of which are integrated with task offloading for CNR in a novel approach to improve the system performance by saving the energy.

Due to all the attributes of virtualization, decentralization and real-time capability, Industry 4.0 is envisioned to be a key area for infusion of these robotic technologies, especially in automating applications such as sensing, actuating and monitoring via insurgence of cloud computing and wireless sensors. In fact, industrial cloud robotics encapsulates the design principle of robotic resources integrated with the cheaper computing cost and network resources, which has extended its operational capabilities and produced a shift in the modes of robotic applications [15] from carrying out repetitive tasks towards solving more complex multi-objective problems in uncertain environments. Our proposed scenario presents one such multi-objective problem where offloading, path planning and AP selection is jointly considered for making the decisions.

Several notable works in the recent past individually studied the topics of path planning, offloading decisions or AP selection rather than a joint study. For example, communication-aware motion [190] and energy-aware coverage path planning for UAVs [191] present works done on robots that utilize surroundings and system information to accomplish goals efficiently. The implementation of Path Planning as a Service [192] provides features of shortest path finding and on-demand planning on the cloud. In the industrial context, path planning studies in industrial applications mostly focus on obstacle avoidance [193] and minimizing the energy [194] for robot-based inspection systems. This led to several studies of mobility-aware offloading or instances where path planning and task assignment/offloading was considered as a joint study, as mentioned in chapter 2. Contrary to these approaches, our proposed work utilizes mobility/path
planning to benefit task offloading to the cloud along with considerations of communications links for task offloading. Since our application space is for CNR, here robots make on-demand movement. So, instead of relying on user mobility patterns or making mobility predictions, a robot plans its path to accommodate offloading for industrial applications of factory maintenance (presented later in the chapter).

Another critical factor for task offloading decision process is communication links (based on AP selection), which is why there are currently several well-established algorithms for improving network throughput and offloading performance by consideration of bandwidth. For bandwidth estimation and allocation, several well-established algorithms have been designed for the purpose of improving throughput by proper AP selection [153] [195]. For example, Li et al. [196] studied the choice of communication link for improved bandwidth/throughput, that led to our cloud-based offloading approach. For the industrial scenario though, AP selection decisions have been studied mostly for specific operations such as efficient energy usage [197] and indoor localization [198]. However, there have been very few instances of research studies where offloading decisions are based on AP selection and its resultant bandwidth, as mentioned in chapter 2 and most of them has been for MCC.

Table 4.1 compares our study to related work (in the areas of motion, AP and offloading) found in the literature and highlights our contributions. In contrast to existing studies, we concentrate on a joint optimization study that considers all three topics. The results in our previous chapter have already showed that offloading can improve system performance by finding optimal offloading decisions for a given application and enhancing the QoS. We also highlighted the relationship of mobility and communication with offloading as well as their impact on offloading decision. In this chapter, we design a GA-based 3-layer decision-making scheme that performs novel three-layer decisions:
(i) whether to offload a task or not, (ii) path planning to reach a desired location for offloading/local execution, and (iii) select access point (AP) for offloading. Since GA is versatile as well as suitable for multi-objective optimization problems in unstructured environments such as ours, we chose to design a modified GA scheme with a novel 3-layer decision for this problem set (over other traditional methods). To the best of our understanding, a modified GA-based method for cloud robotics, jointly optimizing AP selection, offloading and path planning, is the first of its kind. In the following section we propose our mechanism to accommodate a mobility-driven and communication-aware task offloading for a cloud-assisted robotic system.
4.2 Task Offloading Mechanism for Cloud-Assisted Robotic System

Based on our framework presented in chapter 3, we initially propose a task offloading scheme that collectively considers aspects of mobility and communication for its offloading decision-making in a cloud-assisted robotic system. It exploits the interrelation among task offloading, path planning (mobility) and network bandwidth through AP selection (communication). As the mobile robots move on-demand, they may go to intended locations and perform action-based tasks. Due to the inclusion of the cloud, a robot gains computation support for analytical tasks. In the case of offloading, the mobile robot has to share the communication channels with other available users in order to gain its “fair share bandwidth” (based on AP selection) for communication with cloud. Due to the robot’s on-demand mobility, it has the opportunity to choose different points to offload the tasks. Since different locations have different stream rates/throughputs (bit rate) to the APs, the selection of AP and the choice of location interdependently influence the available bandwidth for the robot’s cloud-based communication. In this way, task offloading to the cloud for a robotic agent is heavily dependent on its decision-making that considers motion and connectivity issues. Fig. 4.1 depicts the task offloading mechanism for a cloud-assisted robotic system. The key inter-dependent parameters for robot-cloud communication are:

i) **Offloading Decision**: Depending on the allocation, a task is selected to be either offloaded or not. In the latter case, the task is performed on-board the robot, which doesn’t require any communication with the cloud during the completion of the task. However, for offloaded tasks, the robot needs to communicate with the cloud to perform the necessary computation of that portion of the application.
ii) Movement Decision/Path Planning: Based on the complete decision set, each task needs to be allotted a location where the task will be taking place. As bandwidth for robot-cloud communication is dependent on it (explained later in the chapter), the selection of location helps the robot to plan its path as well as accommodate offloading decisions.

iii) Communication Link/AP Selection: When the task is decided to be offloaded, it requires transfer of data, for which bandwidth is a determining factor of whether it is feasible to offload a task or not. As the robot shares the communication channels with other users, it gains lower bandwidth than the maximum AP throughput. In this case, AP selection helps the robot to gain suitable bandwidth for robot-cloud communication, ultimately resulting in faster offloading and more efficient system performance. This is why AP selection needs to be considered as part of the offloading decision-making.
4.3 Application Use Case: Smart Factory Maintenance

With the recent advances in IoT technology and cloud computing infrastructures, opportunity for innovation arises where automated factory operations in remote locations is run with limited human involvement, increased efficiency and reduced cost. By deploying a pool of heterogeneous wireless sensors throughout the smart factory environment for data collection, we evaluate the processing of environmental conditions with machine learning algorithms (via the cloud) in order to build an operation overview. Based on the information provided, the robotic agents then act upon the environment and complement the sensors with actuation such as scheduled maintenance, fault diagnosis, sensor repair etc. This complements our integrated framework (Fig. 3.2 in chapter 3) for smart factory maintenance where sensors collect and process data, robots assess the information as well as perform computationally heavy tasks and virtual machines (VM) in the cloud provides storage and analytical support.

As a use case, we present an application for automated smart factory maintenance in a remote location [189]. The environment example we consider is one typical to the oil industry, where integrated services provided by WSN and a robotic network complete tedious sets of maintenance-related tasks with little human input. With the shrinking of fossil fuels and the necessity of supplying global demand, newly discovered reservoirs are typically located in areas of extreme environmental and remote conditions, such as hot deserts, deep water and in the arctic zone etc. These pose difficult challenges to health, safety and the environment (HSE) [199]. This has led to the increase in usage of robotic operations for frequent inspection, maintenance and repair (IMR) of plant facilities. Oil facilities have extensive usage of pipes, tanks and heavy machinery that needs constant inspection and maintenance to ensure reliability and continued operation. Semi-autonomous robots have already been used via teleoperation where humans operate these
Figure 4.3: A 30-node application taskflow for smart factory maintenance

robots from safe and comfortable locations, whereas the field robots work as an “extended
physical body” of the operator with various audio, visual and tactile sensors [200].
However, the insurgence of IoT has presented a new possibility for integrating low-cost
WSN and cloud robotics for a fully autonomous maintenance operation in such smart
factories. We present a scenario of an oil factory (as seen in Fig 4.2) where IoT-enabled
sensors detect faults or predict required maintenance so that robots may visit the given location to diagnose the problem and prepare a strategy based on the vast amount of information available through the cloud. The schematic of the workspace is presented in Fig. 4.2. The automated maintenance operation is defined in three stages (Fig. 4.3):

1) **Stage 1: Fault Detection and Localization**: In order to avoid the laborious process of machine maintenance, acoustic monitoring and deep learning technologies for sensors provided by companies such as 3DSignals [201] are used to monitor sensory data (i.e., noise level, temperature, pipe leakage etc.), detect anomalies or identify maintenance requirements. As operations are taking place in a remote location, instead of waiting for the human engineers, a maintenance request from the sensors (WSN) is sent to the nearby service robot ($P_1$) along with the location of the incident for an immediate check-up. As a result, it becomes easier for the robot to localize the fault/problem and update the main centre while reaching that location.

2) **Stage 2: Machine Inspection and Diagnosis**: Based on the information/request from the WSN, the robotic agent provides first on-scene incidence at the notified location ($P_2$) to verify the fault and collect the necessary information to diagnose the fault (Fig. 4.3). This may include performing several different tasks such as wireless communication for sensor testing [202], noise monitoring [203] and analysing X-ray vision of pipes [204], to properly determine the condition of the given section of the machine. On the basis of this information, the robot will analyse a proper strategy and forward it to a human engineer at a remote location who is responsible for performing the proper operation related to problem solution and maintenance. For this part of the operation, the robot may leverage the cloud for computationally heavy tasks as well as update the cloud centre with the necessary updates of the inspection and analytical progress. At the end of this phase,
the robot identifies the main fault reason and awaits maintenance verification (from the engineer) while moving to the location of the main engine.

\textit{iii) Stage 3: Verify Machine Maintenance and Solution:} After collecting the information from the WSN and having diagnosed the problem, the robot analyses the information to suggest the best course of action, which may include changing some parameters of the machine (e.g., temperature, speed, pressure level, valve positioning, operation frequency etc.). Having requested the engineer to make changes to parameters, the robot reaches the location of the main engine ($P_3$) and collects the new parameter information. Concurrently, it also receives feedback from the engineer regarding confirmation of changes or additional changes that needs to be made. After crosschecking all the information, the robot verifies the updated parameter status with the remote engineers. Once completed, the robot sends a final confirmation of task completion to the main centre while moving to its next destination $P_4$ for the next set of tasks.

From proposed scenario, it is seen that the application is time-sensitive, requires a QoS guarantee and results in robot moving to 4 different locations as well as communicating with cloud and WSN. Based on this, we motivate an optimization problem of mobility-driven and communication-aware task offloading in an industrial environment.

\textbf{4.4 System Modelling and Problem Formulation}

Based on the proposed application in section 4.3, we initially model the system that includes the key parameters of mobility, communication and offloading, which results in a joint optimization problem formulation. As previously mentioned, all 3 parameters (i.e., task offloading, path planning and AP selection) are interdependent in the context of the cloud robotic applications. The choice of one of these variables can impact other decisions as well. As a result, our system model takes these factors into consideration [189].
4.4.1 Taskflow modelling and task offloading

As seen, the maintenance application taskflow is a 30-node direct acyclic graph (DAG), presented by a tuple $D = (\mathcal{T}, K)$. Each task is defined as $\mathcal{T} = \{t_j, j = 1:t\}$ and $t = |\mathcal{T}|$. The communication edges between node $v_i$ to $v_j$ are referred by $K = \{k_{ij} = \langle t_i, t_j \rangle\}$ and $k = |K|$. All task nodes are indicated by tasks $t_1 \ldots t_{\mathcal{T}}$. Here the taskflow is divided by levels where task precedence constraint means each task level must be completed in order to start the next one. In addition, the tasks are differentiated into four categories, as seen in Fig. 4.3. These are: must be offloaded (blue), un-offloadable (orange), possible offloading decisions (green) and tasks that require wireless communication (yellow). Here the available heterogeneous resources for allocation are the robot and the cloud VM. Finally, $(P_1 - P_4)$ represent the location constraints for certain tasks as well as help identify the starting point of each stage of the operation.

Figure 4.4: A 36-cell grid derived from the proposed oil factory environment
4.4.2 Workspace and path planning

Based on the factory environment in Fig. 4.2, the workspace is modelled as a grid (Fig. 4.4), where each uniform cell points to a location. The grid-based model is usually used to signify the workspace in the path planning of a mobile robot, as it is easy to calculate distances and represent obstacles. The whole workspace is characterized by orderly numbered grids, and the size of the grids determines how many cells there are. As for the inputs, the cells are defined by \( L, \forall L \in [1: l] \), where the total number of cells, \( l=36 \). Each cell is considered as either empty (available for movement) or occupied (unavailable). Obstacles \( O_i \) indicate occupied cells that are off-limits for movement. These obstacles are adopted from the factory environment and used for simulation later in the chapter. The boundary of obstacles is formed by their actual boundary plus minimum safety distance. In practice, it is performed to consider the size of a mobile robot while moving. Since the robot needs to complete certain tasks (e.g., collect data, read meter etc.) on fixed locations, it is presented as \( T_{ul} = \langle i, l \rangle \); where task \( i \) is constrained to the location \( l \). Finally, the point \( P_1 \) indicates the starting point for the robot and \( P_4 \) indicates the finishing point.

The path and the distance between any two cells in the grid workspace is calculated using a modified A-star method [205], which includes two conditions: \( a) \) the path should be collision-free (with respect to obstacles \( O_i \), \( b) \) and the path should be the shortest distance from a start point to a target point. A-star is defined as the best-first algorithm [152] where the path planning process utilizes the information of distance between the current location and the goal state (target). From its initial position, the robot moves to the cell with the shortest distance from the current location and evaluates cells by combining the distance cost \( h(l) \) to that cell and the distance cost \( g(l) \) to go from that cell to the target. For each successor, the total cost, \( f(l) = g(l) + h(l) \) is then calculated and the cell with the smallest \( f(l) \) is selected as the new successor. By systematically
traversing the whole workspace using this approach, the shortest distance from start to the destination is found. Even though the A-star method could be comparatively time-consuming, it never overestimates the solutions. Therefore, given the relatively small size of the workspace and scope of the work, the implementation of the A-star method is admissible for finding the distance between any two cells in the workspace.

These path and distance values are key parameters in the simulation, because distance coverage impacts the robot’s movement energy as well as its choice of location and AP for task offloading. During the simulation, proposed solutions include tasks being assigned to the location (with respect to the obstacles $O_l$). This helps prepare a robot path plan that accommodates offloading and AP selection for overall performance improvement. Based on the planned movement throughout the application, the total distance $D_{total}$ and movement energy $E_{Mov}$ is calculated as:

$$D_{total} = \sum_{t_i \in m(t)} f(l)_{a,b} \quad (4.1)$$

$$E_{Mov}(t_i, l) = \sum_{t_i \in m(t)} P_{mov} \times \frac{f(l)_{a,b}}{R_v} \quad (4.2)$$

The movement costs for tasks $t_i$ in (4.1) and (4.2) are location-dependent and hence part of the movement set $m(t)$. As mentioned, a modified A-star method is used to calculate the distance $f(l)_{a,b}$ between any two points ‘a’ $(x_1, y_1)$ & ‘b’ $(x_2, y_2)$ in order to eventually get $D_{total}$ and then the movement energy $E_{Mov}$.

4.4.3 Communication model and AP selection

The impact of bandwidth on offloading is thoroughly explored in this chapter by designing the workspace with several APs that allow the users to share the WiFi network and connect to the Internet using the infrastructure mode, as opposed to an ad hoc WiFi network where there are no APs and the nodes directly communicate among themselves.
Since we assume that the communication channels are not dedicated to the robots, the bandwidth is shared with other users. Hence the robot’s share of the bandwidth will be less than the maximum offered by the AP, which is why it is important for the robot to identify the correct AP selection strategy. The communication modelling provides information about users, the number of APs and their locations. Based on this, the robot estimates the “fair-share bandwidth” at different locations as well as plan its path and tasks accordingly in order to gain the most suitable bandwidth (AP selection) for offloading. In this instance, the possibility of the robots being out of a given communication range is ruled out by preparing a workspace where each location is covered by at least one access point. As for the network topology, the robot and the cloud VM maintains a point-to-point connection for wireless transmission of data, which suits the type of application that is proposed in this thesis.

Let’s define an AP by $\alpha$ and total sets of AP as $\mathcal{A}$. The bit rate function is presented by $r(\cdot, \cdot)$ as $r: (\mathcal{L} \times \mathcal{A}) \rightarrow \mathcal{R}$. Here $r(l, \alpha) \in \mathcal{R}$ is bit rate at which the robot individually transmits data from location $l \in \mathcal{L}$ with AP $\alpha \in \mathcal{A}$, where $\mathcal{A} = \{1, \ldots, \alpha\}$. Let $\mathcal{R}$ be the set of bit rates available with the technology being used. Given that set, the individual bit rate $r$ will depend on the location and the AP. The farther the robot is from a given AP, the lower the value of $r$ will be. This bit rate value is known to the robot for any given location during the task operation. Since each access point is shared by a number of users, the bandwidth at a given location is shared among the number of users. The robot’s throughput/bandwidth $\beta$ at location $l$ is estimated according to protocol IEEE 802.11 WLANs [206] as follows:

$$\beta(l, \alpha) = (r(l, \alpha)^{-1} + \sum_{u \in \mathcal{U}_\alpha} c_u^{-1})^{-1}$$  \hspace{1cm} (4.3)
Here $U_\alpha(t)$ denotes the set of users, excluding robot, that are associated with AP $\alpha$. While $c_u$ signifies the cumulative bit rate for the set of users with respect to a given AP $\alpha$. Since, each AP is shared by multiple users and all WiFi users use same packet size, the resulting throughput $\beta(l, \alpha)$ is the “fair-share” bandwidth the robot receives at location $l$, if it selects AP $\alpha$. The throughput equation in (4.3) can be made more detailed to improve accuracy, but in the current form it already captures the essential features of the packet scheduling of the 802.11 MAC in the simplest possible way. Therefore, no further information is provided in this section.

For the given scenario and type of application, the time period is short and hence less sensitive to dynamic changes. As the robot is aware of the total users set $U_\alpha$ and bit rates $c_u$ for each AP $\alpha$, it calculates the bandwidth at given locations (with respect to association with suitable APs). Using bandwidth, the energy for sending instructions $E_I(t_i)$ and uploading data $E_U(t_i)$ during task $t_i$ is calculated for the robot as follows.

$$E_I(t_i) = P_i \times BPI \times \frac{N(t_i)}{\beta(l,\alpha)} \tag{4.4}$$

$$E_U(t_i) = \sum_{t_i \in v(t)} P_u \times \frac{d(t_i)}{\beta(l,\alpha)} \tag{4.5}$$

As seen, $N(t_i)$ is the amount of instruction required to complete the tasks $t_i$ and $E_U(t_i)$ is specifically for tasks that needs to upload collected data $d(t_i)$ (from WSN), hence part of the set containing movement-related tasks $v(t)$. The key parameter in both (4.4) and (4.5) is the bandwidth that is dependent on location $l$ and AP selection $\alpha$. It suggests that AP selection is integrated with the robot’s path plan and offloading decisions, so that robot gains access to better throughput for faster offloading to the cloud, which plays a significant role in minimizing the total energy usage.
Table 4.2: Additional notation (following up from Table 3.1)

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{total}$</td>
<td>Total distance covered by the robot</td>
</tr>
<tr>
<td>$N(t_i)$</td>
<td>Number of instructions for task $t_i$</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Robot processing power for sending instruction to cloud</td>
</tr>
<tr>
<td>$P_u$</td>
<td>Robot processing power for uploading data to cloud</td>
</tr>
<tr>
<td>$P_{cc}$</td>
<td>Robot processing power during cloud computation</td>
</tr>
<tr>
<td>$P_{mov}$</td>
<td>Robot processing power during robot movement</td>
</tr>
<tr>
<td>$R_v$</td>
<td>Robot movement velocity</td>
</tr>
</tbody>
</table>

4.4.4 Optimization problem

Two types of factors are considered for the calculation of energy and latency cost functions (used in derivation): a) fixed parameters—task input, robot and cloud VM processing power; b) variable parameters—offloading, bandwidth and movement. Table 4.2 lists all the additional notations (following up from Table 3.1) of the parameters for energy and latency cost function [189].

i) Cost Function for Energy:

\[
E_{total} = \sum_{i=1}^{[T]} I_{t_i} L_{t_i} E_R(t_i, l) + \sum_{i=1}^{[T]} -I_{t_i} L_{t_i} A_{t_i} E_C(t_i, l, \alpha) \quad (4.6)
\]

\[
E_R(t_i, l) = E_{Mov}(t_i, l) + E_{WSN}(t_i, l) + E_{RC}(t_i) \quad (4.7)
\]

\[
E_C(t_i, l, \alpha) = E_{Mov}(t_i, l) + E_{WSN}(t_i, l) + E_U(t_i) + E_I(t_i) + E_{cc}(t_i) \quad (4.8)
\]
Here robot is the centralized decision-maker and total energy $E_{\text{total}}$ in (4.6) consists of energy from all tasks partitioned into local (on-board) and remote (cloud) allocations. $I_{t_i}, L_{t_i}, A_{t_i}$ are unknown binary variables for the following decisions of each task:

- $I_{t_i} =$ Offloading decision set for each task $t_i$. Here $I_{t_i} (1)$ indicates tasks on the robot. $\neg I_{t_i} (0)$ are tasks offloaded to cloud.
- $L_{t_i} =$ Location for each task where the set consists of total $l$ possible values ($L = 1...l$). For our formulation, $l = 36$.
- $A_{t_i} =$ Selected AP for each offloaded task, where the AP set has a total of $\alpha$ values ($\mathcal{A} = 1...\alpha$). For our problem, $\alpha = 4$.

Depending on selections of $I_{t_i}$, $L_{t_i}$ and $A_{t_i}$, we calculate robotic energy and task completion time. $E_R(t_i,l)$ indicates the total energy for a task being completed on the robot. It includes the movement energy $E_{\text{Mov}}(t_i,l)$, data collection $E_{\text{WSN}}(t_i,l)$ and computation energy $E_{\text{RC}}(t_i)$. As seen from components of the equation, it is dependent on location, hence path planning plays a critical part in offloading decision-making.

Even when a task takes place on the cloud, there is still energy consumed by the robot. This is expressed by $E_c(t_i)$. It consists of movement energy $E_{\text{Mov}}(t_i,l)$ to go to particular location for offloading, energy to collect data from WSN $E_{\text{WSN}}(t_i,l)$, energy to upload data $E_U(t_i)$, energy to send instructions to cloud $E_I(t_i)$ and finally energy consumed while sitting idly during cloud computation $E_{\text{cc}}(t_i)$. All these parameters are either dependent on the location or AP selection. In some cases, it is dependent on both, as seen from the equations. As already mentioned, bandwidth is a determining factor for robot-cloud communication; hence, location (path planning) and AP selection influence the offloading process as well as the offloading decision itself. Finally, the terms $E_{\text{RC}}(t_i)$ and $E_{\text{cc}}(t_i)$ refer to the respective computation energy for a task taking place locally and on the cloud.
VM. For energy and latency calculation, the VM processor clock in the cloud ($S_c$) is considered as M times faster than the robot’s ($S_c = M \times S_r$) processor.

**ii) Cost Function for Time/Latency:**

\[
T_{total} = \sum_{i=1}^{T_i} I_{t_i} L_{t_i} T_{R}(t_i, l) + \sum_{i=1}^{T_i} L_{t_i} A_{t_i} T_{C}(t_i, l, \alpha) \tag{4.9}
\]

\[
T_R(t_i) = T_{Mov}(t_i, l) + T_{WSN}(t_i, l) + T_{RC}(t_i) \tag{4.10}
\]

\[
T_C(t_i) = T_{Mov}(t_i) + T_{WSN}(t_i) + T_U(t_i) + T_{CC}(t_i) + T_I(t_i) \tag{4.11}
\]

In (4.9), $T_{total}$ is the total task completion time, whereas $T_R(t_i)$ and $T_C(t_i)$ are the completion times on the robot and the cloud. From this point onwards, the rest is calculated similarly to the energy calculation.

**iii) Optimization Problem:** Based on the problem of the proposed application (Fig. 4.3), the objective is to find near-optimal decisions for offloading ($I_{t_i}$), path planning ($L_{t_i}$) and AP selection ($A_{t_i}$) together within the constraint (latency) imposed, thus providing communication-aware and mobility-driven task offloading which results in minimum energy consumption for the robot.

**Scenario:** Minimize robotic energy for constrained latency [189].

Find: $\{I_{t_i}, \{L_{t_i}\}, \{A_{t_i}\}, \forall \mathcal{J} = \{v_j, j = 1: t\}$ and $t = |\mathcal{J}|$ to

\[
\text{Minimize: } E_{total}
\]

\[\text{s.t.: } T_{total} \leq T_{Deadline}\]

In the following section, a modified GA scheme with a novel 3-layer chromosome/solution is designed to solve the joint optimization problem.
4.5 GA-based 3-Layer Decision-Making Scheme

As previously mentioned, GA is a successful evolutionary computational intelligence branch, where weak and unfit species are eliminated by natural selection and stronger genes are passed on to the next generation by reproduction [67]. This heuristic approach works efficiently for large scale multi-objective optimization problems, because it approximates brute force without enumerating all the elements, thereby bypassing performance issues specific to an exhaustive search [207] which suits the NP-complete problem set in this work. In the field of evolutionary robotics, GA has generally been implemented in order to optimize the control policy of a robot [115]. This technique is used in applications to rapidly locate the “satisficing” solutions when sufficient priori knowledge is not available.

Previously, path plan, offloading or AP selection have been individually studied as examples of classical machine learning problems being solved by adaptive learning without significant domain knowledge. However, with the rapid increase in technology, the complexity of handling dynamic and multifunctioning systems is exponentially increasing because of factors such as dependencies among parameters, difficulty to map, interconnections etc. In order to avoid situations where certain aspects of development may become “intractable” due to constant progress and evolution in response to progressive conditions and demands, it is of utmost importance to prepare more comprehensive techniques to model systems to deal with dynamic changes and high levels of complications. Therefore, the interdependent parameters (path plan, AP selection) are integrated with offloading decisions in this chapter and a modified GA scheme with novel 3-layer decision-making strategy is designed in order to minimize $E_{total}$ [189]. Given the nature of the problem, there aren’t many analytical or traditional
algorithms in the literature to solve the problem in an efficient way. Due to the unstructured type of problem presented in this chapter, GA is the most suitable approach as the algorithm can adapt accordingly. In the context of the problem, the algorithm is trained and driven towards an area of optimal/near-optimal result with high probability. In order to design a GA-based scheme, the following steps needs to be followed, wherein modifications are required in order to cater to the needs of the application:

i) Three-Layer Modified Chromosome Encoding: The process initiates with a modified population (P) that is randomly generated with a collective unit of novel 3-layer chromosomes (offloading, path location, AP) from the search space. It is presented as 

\[
P = [I_{t_1}, I_{t_2}, \ldots, I_{t_T}, L_{t_1}, L_{t_2}, \ldots, L_{t_T}, A_{t_1}, A_{t_2}, \ldots, A_{t_T}].
\]

An example is presented in Fig 4.5 to further explain the process. As seen in the figure, \( I = [0\ 1\ 2\ 4\ 3\ \star] \) suggest task 1 is completed on cloud VM (0) by offloading at location 2 with AP 3. Whereas, task 2 is performed by robot (1) at location 4 without any AP. The task graph in Fig. 4.3 highlights allocation constraints—local task (orange), WSN (yellow), cloud communication (blue) and offloadable tasks (green). In addition, \( P_1 - P_4 \) are location constraints. This means locations for some tasks are pre-defined based on their type (e.g., collect data, machine inspection etc.). For the same reason, their allocations are fixed as well (on the robot or the cloud). At this instance, the encoded chromosome for each population is further modified with a fixed allocation and location \( (I_{t_i} \& L_{t_i}) \) in order to accommodate these constraints and provide a real-world context for such tasks (e.g., collect data, investigate). A sample has been in shown in Fig. 4.5.

ii) Fitness Parameter Calculation and Evaluation: Chromosomes/solutions in the population are evaluated by a fitness function to measure the performance of the proposed solution in the search space. For the given problem, the fitness measure considered is 

\[
f = E_{\text{total}}.
\]

The objective is to find lower values of energy and replace previous ones to
ultimately find the lowest energy. In the context of this problem, “lowest energy” indicates “higher fitness measure”. Accordingly, the ultimate result will provide the lowest energy, which is the best fitness measure. For the calculation of parameters i.e., total robotic energy ($E_{\text{total}}$), delay ($T_{\text{total}}$) and distance ($D_{\text{total}}$), equations (4.1)-(4.11) are used. Finally, a breadth-fast search is performed to identify task dependencies and to divide them into groups for a level-wise calculation. A flowchart in Fig. 4.6 explains the calculation process of fitness measures and other performance parameters.

Based on the chromosome allocation, we calculate the values for $E_R(t_i, l), E_C(t_i, l)$ in accordance with the corresponding equations to get $E_{\text{total}}$ for a single level. As for distance, this is calculated according to the third layer of the chromosome, which is path planning ($L_{t_i}$). For every task that is part of the set $m(t)$, distance cost between the corresponding points is calculated (via A-star method). Since, all energy and movement costs are additive, $E_{\text{total}}$ (fitness measure) and $D_{\text{total}}$ from each task level is calculated sequentially and added to ultimately get the respective values of $E_{\text{total}}$ $D_{\text{total}}$.
Input: Workspace, DAG, $I_{ti}$, $L_{ti}$, $A_{ti}$

Output: $E_{total}$, $T_{total}$, $D_{total}$

1: Initialize $E_{total}$, $T_{total}$, $D_{total}$
2: for each level $H_j \in H_{total}$ do /* Calculate energy, time, distance*/
3:     for each task $t_i \in H_j$ do
4:         $H_j(l_e) \in \{0,1\}$ /*Find task allocation*/
5:             $H_j(L_{ti}) \in L = \{1, \ldots, l\}$
6:                 $H_j(A_{ti}) \in A = \{1, \ldots, a\}$
7:         if $t_i \in m(t)$ /*movement task set*/ do
8:             $E_R(t_i, l) = E_{Mov}(t_i, l) + E_{WSN}(t_i, l) + E_{RC}(t_i, l)$
9:         else do
10:             $E_R(t_i, l) = E_{WSN}(t_i, l) + E_{RC}(t_i, l)$
11:         end if
12:     end if
13: end for
14: end for
15: if $l_{ti}(1)$ /*task on robot*/ do
16:     if $t_i \in m(t)$ /*movement task set*/ do
17:         $E_C(t_i, l, \alpha) = E_{Mov}(t_i, l, \alpha) + E_{WSN}(t_i, l, \alpha)$
18:             $+E_U(t_i, l, \alpha) + E_I(t_i, l, \alpha) + E_{cc}(t_i, l, \alpha)$
19:     else do
20:         $E_C(t_i, l, \alpha) = E_{WSN}(t_i, l, \alpha) + E_U(t_i, l, \alpha) + E_I(t_i, l, \alpha) + E_{cc}(t_i, l, \alpha)$
21:     end if
22: end if
23: end if
24: end for
25: if $T_R(t_i, l) > T_C(t_i, l, \alpha)$ do
26:     $T_{total} = T_{total} + T_R(t_i, l)$
27: else do
28:     $T_{total} = T_{total} + T_C(t_i, l, \alpha)$
29: end for
30: end for
31: Calculate $E_{total}$, $T_{total}$, $D_{total}$

Figure 4.6: Pseudocode for robotic energy, time and distance calculation
Since the tasks on the same level happen in parallel, total time $T_{total}$ for tasks is not directly additive. Therefore, a slight modification is required to compensate for parallel tasks. For each level, we divide tasks into two types based on allocations (0 or 1) and put them into two different lists. We then calculate the cumulative time for both lists and use the higher value between the two as the eventual latency/time cost from that DAG level. This is then added to the overall results to get the total task completion time $T_{total}$.

![Figure 4.7: Crossover phase for GA-based offloading scheme](image-url)
iii) **Selection Phase:** After the fitness calculation, the mating pool is filled in iteratively from the current generation. Two chromosomes are randomly selected in each pass and the individual with higher fitness measure (i.e., low energy) is finalized to fill the mating pool. The process is repeated until the mating pool is completely filled in. We also adopted elitism by keeping the historical best solution in the mating pool.

In addition, infeasible solutions are dealt with at the fitness calculation stage. These solutions are given very low fitness measures (i.e., infinite energy scores). By so doing, they are never be picked up for the mating pool (as their fitness will be always inferior to feasible solutions).

iv) **Crossover Phase:** Similar to the crossover phase in chapter 3, two selected chromosomes from the previous phase “reproduce” in the crossover section and produce “offsprings”. Here the “uniform crossover” process is considered as it uses a fixed mixing ratio between the two parents. This process allows the parent chromosomes to contribute at the gene level rather than at the segment level. However, instead of one chromosome layer, all three layers of chromosome “crossover” to produce child chromosomes (Fig. 4.7). During this stage, individual bits in the string are compared between their two parents and, all the bits are swapped with a fixed probability.

As mentioned, infeasible solutions are removed by giving them low fitness measures in order to avoid degradation of GA performance. However, if any such solution is still produced at crossover, it would be eliminated at the fitness calculation and mating pool generation stage of the following pass of the GA.
Figure 4.8: Mutation phase for GA-based offloading scheme

v) **Mutation Phase:** At the end of selection and crossover phase, there is now a new population full of possible solutions (decision set). However, the chromosomes may become too similar to each other in some cases. At this point, the mutation phase is performed, in which a portion of new individuals have some of their bits flipped with low
probability. This is done intentionally so as to ensure proper diversity and the possibility of finding a global optimum in the search-space. In contrast to the process in chapter 3, all three layers have their bits flipped randomly from their set of possible values (as seen in Fig. 4.8). As a result, we obtain a new population of individuals and the same process is continued. However, any infeasible individuals (in all three layers) are excluded in same way as is done in the “crossover” phase.

vi) **Self-Stopping Criteria:** A self-stopping criteria is embedded, so the process doesn’t evoke unnecessary latency or processing power. Since our GA scheme provides the “lowest energy” (best fitness measure) after each generation, it will only stop when there is no change in the best fitness measure for a prefixed (determined by user) number of generations. At that point, GA is terminated immediately, and the result is considered to be near-optimal.

### 4.6 Simulation Results and Analysis

Extensive simulations are run to evaluate the performance of our GA scheme with variable movement (GAVM). In order to do so, we compare our results with offloading results from a GA scheme with fixed movement (GAFM) (designed in chapter 3) that has already been validated as near-optimal results with respect to exhaustive solutions. In GAFM, the values of movement and communication are presumed to be fixed whereas in this method, we consider them as variable parameters that are part of the decision-making process. In addition, an All-on-Fixed-Resources (AoFR) approach is used as a benchmark where all tasks are allocated to the robot (with fixed location), except for mandatory cloud communication (e.g., updating main centre about maintenance work), that also has fixed location and AP.
Table 4.3: Parameter setup for simulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (Min: $E_{total}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline</td>
<td>Time (500 sec)</td>
</tr>
<tr>
<td>Population</td>
<td>300</td>
</tr>
<tr>
<td>Stopping</td>
<td>1000 generations without change in fitness measure</td>
</tr>
<tr>
<td>Task Constraints</td>
<td>Task (3,4,6,8,9,15,16,18,20,22,27,28,30) on robot</td>
</tr>
<tr>
<td></td>
<td>Task (1,2,12,14,23,25,26) on cloud</td>
</tr>
<tr>
<td></td>
<td>Task (7,19)-WSN</td>
</tr>
<tr>
<td>Obstacle Cells</td>
<td>Cell 3,4,14,16,18,20,22,24,33,34</td>
</tr>
<tr>
<td>No. of AP</td>
<td>4 APs</td>
</tr>
<tr>
<td>No. of Users</td>
<td>Each AP is associated to 3 users</td>
</tr>
<tr>
<td>Location Constraint</td>
<td>Task Zone</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
</tr>
<tr>
<td></td>
<td>6,7,8,9 32</td>
</tr>
<tr>
<td></td>
<td>Task Zone</td>
</tr>
<tr>
<td></td>
<td>18,19,20,22 5</td>
</tr>
<tr>
<td></td>
<td>30 30</td>
</tr>
</tbody>
</table>

The 30-node taskflow for simulation is motivated from the proposed application in Fig. 4.3. Table 4.3 presents all the configuration parameters. For the purpose of this simulation, an Intel Core i5-4570 ($BPI = 3$) processor is considered as the local machine (robot). In addition, the robot clock speed ($S_r$) is 5 GHz. Similar to the previous simulation (chapter 3), the cloud VM processor clock speed, $S_c = M \times S_r$. Processing power ratings are $P_r=75$ W, $P_t = 35.5$ W, $P_{cc}=10$ W, $P_u =50$ W, $P_m=50$ W and $P_d=35$ W. Finally, the workspace is adopted from the previously proposed factory model and is simplified in Fig. 4.4 for the illustration of results. As seen, for communication modelling, an infrastructure using IEEE 802.11 WLANs is presented with a maximum bit rate of 54 Mbit/s and eight available stream bit rates (6,9,12,18,24,36,48,54 Mbit/s). Based on this,
a 36-cell workspace is designed to have four APs (with additional users). Moreover, Table 4.3 highlights workspace obstacles the robot can’t move to. As for the robot, certain tasks have been constrained to fixed locations, which is also detailed in Table 4.3. All this information is available to the robot at the time of operation. Based on this, the simulation results for our mobility-driven and communication-aware offloading (GAVM) are presented as follows:

Table 4.4: Performance of GA-based decision-making scheme

<table>
<thead>
<tr>
<th>Line 1: Task No. (1-30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1 (Line 2): Offloading Decision (1-Robot) (0-Cloud)</td>
</tr>
<tr>
<td>Layer 2 (Line 3): Location (1-36)</td>
</tr>
<tr>
<td>Layer 3 (Line 4): AP (1-4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>21</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>×</td>
<td>×</td>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>23</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>×</td>
<td>4</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>4</td>
<td>×</td>
<td>×</td>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

4.6.1 GA scheme decision-making performance

Simulation results are presented in Table 4.4, where decision types are presented in three layers. The first line is the numbering of tasks. Layer 1 in line 2 (yellow) is offloading decisions for task 1-30, represented serially by 1-Robot, 0-Cloud. Layer 2 in line 3 (green) is path planning, as shown in Fig. 4.9. Here dotted lines indicate offloading locations and continuous lines suggest intermediate paths. Finally, Layer 3 in line 4 (grey) suggests AP selections to offload corresponding tasks with ‘×’ indicating no AP for the un-offloadable ones (local task).
Figure 4.9: GA-based path planning results under normal scenario

Figure 4.10: Fitness performance for GA scheme (Min: $E_{total}$)

In addition, performance for GAVM (via average fitness measure) in Fig. 4.10 depicts a declining trend. This happens because GAVM looks for lower energy ($f=E_{total}$), which is considered as a higher fitness measure. According to the graph, the lowest energy value gets updated with any new lower score and ultimately results in minimum energy (Min: $E_{total}$), which is the best fitness measure. Simulation results from mobility-driven and communication-aware task offloading (GAVM) are validated with respect to other algorithms (i.e., GAFM & AoFR).
Table 4.5: Performance comparison for offloading (Min: $E_{\text{total}}$)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>GAVM</th>
<th>GAFM</th>
<th>AoFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation</td>
<td>6788</td>
<td>2524</td>
<td>N/A</td>
</tr>
<tr>
<td>Offloading Decisions (1-Robot)</td>
<td>0 1 1 1 1 0 1 1</td>
<td>0 0 1 1 1 1 1 1</td>
<td>0 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>(0-Cloud)</td>
<td>1 0 1 1 1 1</td>
<td>1 0 0 0 0 0 1 1</td>
<td>1 1 1 0 1 0 1 1</td>
</tr>
<tr>
<td>Offloaded</td>
<td>11 tasks</td>
<td>9 tasks</td>
<td>6 tasks</td>
</tr>
<tr>
<td>Min. Energy</td>
<td>13442.52 J</td>
<td>15172.37 J</td>
<td>33736.42 J</td>
</tr>
<tr>
<td>Time</td>
<td>281.31 sec</td>
<td>245.01 sec</td>
<td>531.28 sec</td>
</tr>
<tr>
<td>Distance</td>
<td>251.42 m</td>
<td>220.71 m</td>
<td>220.71 m</td>
</tr>
<tr>
<td>GA Overhead</td>
<td>39.2 sec</td>
<td>6.4 sec</td>
<td>N/A</td>
</tr>
</tbody>
</table>

4.6.2 Offloading performance comparison

From a comparison with GAFM and AoFR in Table 4.5, it is evident that GAVM results in lower minimum energy of 13442.52 J w.r.t. GAFM (15172.37 J). Even though latency for GAVM is 281.31 sec (higher than 245.01 sec for GAFM), it is still within constraint. So, the application is completed within time due to the higher number of offloaded tasks (11 tasks), because the robot covers more area (251.42 m) and selects better APs to offload. In contrast, performance of GAFM is limited by fixed movement (220.71 m), fewer offloading (9 tasks) and higher energy usage. Since GAFM has already been established as a near-optimal solution (in chapter 3), the results of the proposed GAVM (w.r.t. GAFM) are shown to have provided near-optimal results as well.

As for AoFR, all the tasks are fixed to the robot, except for the mandatory cloud communication tasks (six tasks) that have fixed/constrained locations for movement and AP. Therefore, this lack of adaptability results in higher energy (33736.42 J) and missed
deadline (531.28 sec). It indicates that the cloud computing infrastructure and the resulting possibility of manipulating the resources for task offloading decisions greatly increases efficiency for robotic applications.

The aforementioned results indicate that mobility-driven and communication-aware offloading (GAVM) provide near-optimal solutions and improves system performance by outperforming AoFR in terms of adaptability for resource usage and GAFM technique in terms of a tighter deadline and faster offloading. Furthermore, the simulations in chapter 3 and 4 indicate that GA performs well and finds optimal results in different complexities of task graphs. Even though, more sets of decisions has been added, GA still manages to find the optimal result. However, due to added layers of decision-making, it takes more time now to find suitable solutions (39.2 sec), as compared to our approach from chapter 3 (6.4 sec). A follow up simulation in chapter 5 will tackle a more complex task graph to analyse the performance of a GA based scheme.

![Figure 4.11: Energy performance of energy-distance weighted sum for fitness score (with respect to initial battery life)](image)
4.6.3 Energy-distance weighted sum for fitness score

To achieve a proper load balance and desired outcome, the interrelation between energy and distance is further investigated by presenting the fitness score as weighted sum values. This allows the robot to control how much effort is put on movement in relation to energy. This new fitness is integrated with initial battery life of the robot’s CPU in order to get an indication of performance changes w.r.t to battery capacity.

\[ f = E_{\text{total}}^\omega \cdot D_{\text{total}}^{(1-\omega)}; \quad \forall \omega = \{0, 0.5, 1\} \]  \hspace{1cm} (4.12)

Combined fitness with \(\omega\) as a weighing parameter means that near-optimal decision-making considers the relative significance of both energy and distance for decision-making. Three values of \(\omega\) (0, 0.5, 1) indicate three battery life stages: low–below 30% (\(\omega = 1\)), medium–30% to 70% (\(\omega = 0.5\)), and high–above 70% (\(\omega = 0\)). Results in Fig. 4.9 and 4.10 clearly outline the performance for weighted sum offloading with respect to battery life. According to our results, for initial battery life <30%, the focus is to minimize energy. Hence it has the lowest energy among the three cases. In contrast, the minimum
energy is higher for battery life >70%. Since, more energy is available in this case, the robot’s primary objective is to minimize movement. Finally, for medium battery life the system performs well-balanced, as both energy and distance are close/equal to minimum (near-optimal). All these results highlight the robot’s adaptability to the changing conditions (initial battery life) and parameters (energy, distance).

(a) Path planning with recharging after part 1 (initial battery < 10%)

(b) Path planning with recharging after part 2 (initial battery < 20%)
(c) Path planning with recharging after part 3 (initial battery < 30%)

Figure 4.13: Visual representation of a 36-cell workspace for an oil industry with recharge-based path planning results (parts a, b and c)

Table 4.6: Robot performance for recharge-based offloading

<table>
<thead>
<tr>
<th>Time of Recharge</th>
<th>Battery Life</th>
<th>Energy</th>
<th>Distance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>After part 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(task 1-5)</td>
<td>Less than 10%</td>
<td>15120.12 J</td>
<td>477.69 m</td>
<td>377.91 sec</td>
</tr>
<tr>
<td>After part 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(task 6-17)</td>
<td>Less than 20%</td>
<td>14521.37 J</td>
<td>284.85 m</td>
<td>307.41 sec</td>
</tr>
<tr>
<td>After part 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(task 18-29)</td>
<td>Less than 30%</td>
<td>16835.50 J</td>
<td>422.84 m</td>
<td>456.93 sec</td>
</tr>
</tbody>
</table>

4.6.4 Recharge-based offloading and path planning

As robots have hardware constraints, recharging the batteries is required to increase performance longevity. Since the robot can check its initial battery status, it utilizes this information to choose the right time to recharge (R) during application, as long as it meets the required application criteria. As seen in the previous section, it is clear that the robot adapts with relation to battery capacity. Therefore, in this section, three stages of
application are considered for recharging, depending on their corresponding battery life:

\( a) \) after part 1 (task 1-5) for initial battery life \( \leq 10\% \), \( b) \) after part 2 (task 6-17) for initial battery life \( \leq 20\% \), and \( c) \) after part 3 (task 18-29) for initial battery life \( \leq 30\% \). Here the idea is for the robot to complete one part and plan its path towards the next part by going through the recharge centre. Additional latency for recharging \( (R) \) is also considered for \( T_{total} \) calculation, with simulation results presented in Fig. 4.13. As seen, Fig (a), (b), (c) are path planning results with recharging, hence the respective values for total distance is relatively higher \((477.69 \text{ m}, 284.85 \text{ m}, 422.84 \text{ m})\) than the general scenario \((251.42 \text{ m})\), as seen in Table 4.6. In addition, latency of 377.91 sec, 307.41 sec, 456.93 sec are also high, even though within constraint \((500 \text{ sec})\). Finally, the energy of 15120.12 J, 14521.37 J and 16835.50 J also indicates lower values of energy, even though they are comparatively higher than the minimum energy. This is because of the additional paths the robot takes for recharging, which evokes extra energy. However, all the results are reasonably low and within constraints. Hence, they are considered as possible solutions in these particular cases where recharging is involved. In the context of the maintenance application, this ability adds flexibility to the robot as it may recharge while completing a task set instead of delaying an operation due to recharging issues.

### 4.7 Summary

In this chapter, a novel 3-layer solution (task offloading, path planning, AP selection) is designed to leverage the complementary strength of the robot’s on-demand mobility by jointly considering offloading, network communications and movement decisions for a cloud-assisted robot-based system. The 30-node taskflow and 36-cell workspace represent a smart factory maintenance application for our integrated framework of WSN and cloud robotics. Next, a joint optimization problem is formulated, followed by the
design of a modified GA-based decision-making scheme (with novel 3-layer decision set) to find the near-optimal solutions. According to results, the modified GA scheme achieves minimum energy for the given application and finds the following decisions: near-optimal task offloading, path planning and AP selection. As the outcome suggests, communication-aware and mobility-driven task offloading improves the performance in comparison to existing validated techniques (with fixed parameters). It also presents the opportunity to further utilize this relationship to balance the load between movement and energy along in order to make the system more adaptive to changing conditions. Furthermore, we also present a scenario where the robot plans to “recharge itself” in the middle of an application while accommodating the offloading decision-making, as long it meets the application constraints.

To summarise, a mobility-driven and communication-aware offloading scheme for a cloud-assisted robot-based system lead to superior system performance. Unfortunately, it is only suitable for applications with single robot activities. However, for a multi-robot operation, there are additional considerations to be made. Consequently, in the following chapter, we will address the multi-robot scenario and design an offloading scheme specifically for multi-robot cloud networked systems.
Chapter 5.

Energy-Efficient Optimal Task Offloading for Cloud Networked Multi-Robot Systems

Task offloading in multi-robot cloud-assisted systems is multifaceted (contrary to single robot) and more complex, as there exists an added aspect of robot-robot communication along with the robot-cloud communication. In order to overcome these complications, this chapter aims to design a novel multi-layer task offloading decision-making scheme specifically for multi-robot systems, which jointly considers four aspects: mobility, communication, local robot-robot sharing and robot-cloud sharing. Based on our integrated framework, we consider a smart warehouse scenario application of “parcel sorting and distribution” where the offloading decision for each task is formulated as a joint optimization problem, but now it is solved by designing a modified GA scheme with 4-layer decision set. In contrast to the previous chapter, the additional layer points to other available robots that aid the primary robot to offload tasks via local communication. This is highlighted in the simulation outcome that depicts a significant performance improvement for multi-robot systems due to the involvement of local robot-robot communication on top of the mobility-driven and communication-aware offloading, which facilitates energy-efficient completion of tasks and better utilization of resources.

5.1 Introduction

Multi-robot task allocation (MRTA) is a classical yet complex problem in the field of artificial intelligence (AI). In applications where single robot may struggle to effectively operate and complete tasks in highly complex environments, a team of robots has the capability of distributing the computational load and operating in these environments
more effectively. Especially the addition of the cloud presents the ideal opportunity to maintain more resource-hungry and time-consuming operations by proper coordination among the resources. Therefore, the cloud networked multi-robot applications nowadays generally operate with cooperative control that adopts a decentralized approach to avoid a single point of failure and perform tasks with reduced energy and in less time/delay.

This is especially visible in the industrial realm, where the technical advancement in wireless network technology [208], the Internet of Things [209] and cloud computing have given rise to more progressive networked robotic systems. As a result, these applications have now moved beyond their traditional deployment in production lines and begun to deal with new challenges of industrial applications (e.g., negotiation-based decision-making, dynamic environmental disruptions, human-machine interaction and more personalized consumption demands). One of the most common and well-studied industrial robotic applications is “Warehouse Logistics” where customized ordered parcels are traditionally sorted and distributed with the help of human labour and support from heavy machinery. For smart factories though, the inclusion of interactive cloud-aided robots with advanced communication technology produces a shift in the modes of application from carrying out repetitive tasks towards performing dynamic tasks that requires robots to solve complex multi-objective problems, thus playing a pivotal role in design and management of smart warehouses.

There have been numerous studies for multi-robot platforms on the design of automatic warehouses [210], its multi-robot functions (e.g., task assignment [155], coordination, path planning [158], speed improvement) as well as its various range of applications (disaster management [211], automated order processing [172], assembly cell control [212] etc.). With regard to cloud networked robotic systems, some notable studies have put emphasis on manufacturing applications [106], maintenance-related tasks [213] and
computer vision. As mentioned in chapter 2.5.3, several studies have concentrated on cloud-robot collaborative aspects of industrial applications. However, cloud-aided automated robotic approaches for warehouse logistics and studies related to these are few and far between. Some conceptual work by Bonkenburg has suggested the possible ways where robots can be used in the environment of smart warehouses [214]. Our work is partially motivated by this concept, whereby we consider a warehouse application for automated parcel sorting and distribution. However, the emphasis of our work is on the mobility-driven and communication-aware offloading for a cloud-aided multi-robot system. This motivates our joint optimization formulation where task offloading decisions are presented as an allocation problem among multiple robots and the cloud, which is then solved by a novel genetic algorithm-based (GA) multi-layer decision-making scheme.

As previously explained, task offloading is one of the major benefits of cloud computing where computation-heavy and resource hungry tasks are migrated to a remote yet powerful cloud server for execution. Since the ubiquitous resources of the cloud is rapidly provisioned and released with minimal service provider interaction or management effort, cloud computing allows the energy-constraint robot to offload a portion of the computation to the cloud in order to potentially reduce task execution time and energy. In such a scenario, it is important to identify the appropriate tasks to offload to the cloud, as it may depend on the constraints as well as the type of tasks and objective of the application. As previously highlighted, task offloading for robotics is a particularly complex issue due to their on-demand mobility and network connectivity that significantly influence the robot-cloud communication links. This is why both path planning and AP selections have already been considered as part of their decision-making (in the previous chapter). Furthermore, for multi-robot systems, we simultaneously need
to consider two additional key issues: (a) local resource sharing (robot-robot), and (b) offloading to the cloud (robot-cloud). The added feature of local sharing among the robots has the potential of further minimizing the system energy by distributing the principle robot’s workload to the other available robots in the system either for computational support or to help with task offloading.

Therefore, our method in this chapter considers four layers (e.g., task offloading, robot selection for offloading, task location and AP selection) of decision-making for each task. We believe that by utilizing the available robots from the multi-robot systems, the offloading process can be hastened which enhances the system performance. At the same time, the tasks can be completed with less energy consumption. A thorough simulation has been run, the results of which are presented later in the chapter to identify the offloading decisions for a joint optimization problem based on the 40-node smart warehouse application of “parcel sorting and distribution”. A GA-based decision-set is then designed that focuses on the four key decisions for each task: (i) selection of task for offloading, (ii) selection of robot to offload a task, (iii) selection of location to offload/perform task, and (iv) selection of access point for offloaded task. The results of the simulation are compared with our mobility-driven and communication-aware offloading application from the previous chapter where a single robot has been considered for task completion. The main purpose of this chapter is to highlight the benefit of an optimal allocation scheme among multi-robots and the cloud, which motivates the primary robot to benefit from the support of available robots in the application scenario. This in turn brings about easier offloading and faster completion of tasks with a lower consumption of overall robotic energy.
Figure 5.1: Task offloading mechanism for cloud networked multi-robot system with collective consideration for mobility, communication and local sharing

5.2 Task Offloading Mechanism for Multi-Robot Systems

Based on our integrated framework (Fig. 3.2), we propose a task offloading scheme in a cloud networked multi-robot application (as seen in Fig. 5.1) that collectively exploits the interrelation among offloading, path planning (mobility), bandwidth (communication) and local robot-robot (R-R) sharing. As previously stated, the on-demand mobility allows robots to plan their paths in accordance with their choice of communication link (AP selection) in order to offload tasks to the cloud for computation support. Moreover, for
multi-robot systems, there is an added dimension of local robot-robot (R-R) communication. Even though the added dimension makes the process more complicated, it trades-off in terms of gaining better offloading through proper utilization of local available robots. Hence, the offloading scheme for a cloud networked multi-robot system is divided into 2 types depending on the respective decision of offloading for each task.

i) **Cloud-based Task Offloading (Robot-Cloud):** Depending on the allocation, when a task is selected to be offloaded to the cloud, the robot needs to communicate with the cloud infrastructure via network layer APs. In such cases, the following decisions are considered: a) *which task to offload*, b) *which robot to offload*, c) *which location to offload from*, d) *which access point to select for offloading*.

ii) **Local Task Offloading (Robot-Robot):** Local task offloading occurs in two scenarios. When a task is offloaded to the cloud, the choice of a separate robot from previous tasks enables local offloading (robot-robot). The secondary robot collects task-related information from the primary robot and then transfers this to cloud. This may happen when the primary robot has parallel tasks to complete or the secondary robot is in a better location to offload to the cloud. As for the case when tasks are not offloaded, they take place on-board the allocated robot or are locally offloaded to another robot. This also requires primary robots to share information with the other robots, once one task is finished. In both cases, primary robot is centralized decision-maker and all information is shared among the robots through the principle robot, which suits the type of application proposed in this study. As robots have the capacity to form local networks in an ad-hoc manner, the robots use the network to transfer information to other nearby available robots (within range) for task completion. In this case, the decision-making (Fig. 5.1) may include: a) *which task to take place on a robot*, b) *which robot to complete the task*, c) *whether to locally offload the task to another robot*. 
5.3 Application Use Case: Smart Warehouse Parcel Management

The widespread innovation in robotic technology and subsequent increase in their computing capabilities (due to the addition of the cloud) are enabling their usage in different automated industrial applications. The impact of such advancement is being reflected in the warehouse applications as well, where more operations are now moving towards running with automation support. According to recent reports, around 15% of current warehouses are mechanized. Even though 5% of the warehouses are automated,
most of them are typically mechanized environments that still employ people in key functions. It suggests that there is room for implementation of automated components, i.e., robotic agents. Having initially started with approaches such as teleoperation and later upgraded to automation, the service robots have now reached the age of CNR, where cloud computing in robotic applications has made its significant mark in the industrial realm by enhancing robot operations via on-demand computation and storage support.

In this way, the integration of networked robotic systems, IoT enabled sensors and the cloud infrastructure has led to intelligent perception and on-demand shared resources [169] in the industrial scenario. Such implementation has resulted in several automated warehouse applications for material handling including conveyers, sorters, goods to picker solutions and other mechanized equipment that has the potential to improve the productivity of the existing workforce [214]. In this context of our application, a pool of wireless sensors is deployed in static warehouse machineries (e.g., goods packing, labelling etc.) for data collection and environmental monitoring so that they gain knowledge on the overview of the application. These wireless sensors are complemented by several dynamic robotic agents that move on-demand to perform object pick-up, delivery and drop-off. The integration of these cyber-physical components and wireless sensors enables proper communication over networks for data-sharing and the automated processing of operations that start from the production line and proceed all the way to delivery. Since the design and operation in industrial operations involve numerous varieties of decision-making [171], the inclusion of cloud computing makes an integrated framework of networked robots, the sensors and the cloud that reconciles conventional warehouse problems and perform applications in a semi-automated manner with minimal human supervisory oversight, increased efficiency and more safety and speed.
Figure 5.3: Details of an automated parcel sorting and distribution application in a smart warehouse
(a) 40 node task Graph

(b) Task constraints of robot $R_1$ at different locations
In this chapter, we present a “parcel sorting and distribution application” in an automated warehouse environment (Fig. 5.2). Being automated in nature, the application deals with five major steps that require a mobile robot to complete a set of tasks necessary to prepare a parcel for delivery. Given our proposed application type, each task can be completed by any of the robots, to be decided by the primary robot as part of its decision-making. However, the offloading can be done using other robots as shown in chapter 5.2. For our multi-robot system, we incorporate the principal robot to complete the major actuation-based tasks through interactions with different types of agents (each with a specific job to help with tasks such as unloading objects from trucks, co-packing, picking orders, checking inventory or shipping goods). Hence, the principle robot is the centralized decision-maker that communicates with all the other robots. The supporting robots provide analytical and computation support while completing their own set of
application-related activities. Through this multi-robot communication, the principal robot transfers tasks locally (robot-robot) to other available robots and complete them on-board. Alternatively, it may also get help from the supporting robots regarding offloading a task to the cloud for utilizing its ubiquitous resources. In this way, the other available robots in this shared framework works as a hub to help with local or cloud-based offloading of tasks (if required), whereas the principal robot carries out fundamental aspects of the warehouse management application. As seen in Fig. 5.4b, the complete warehouse-based application is divided into four major steps that involves the primary robot visiting five different locations.

i) **Parcel Request Generation Phase (Stage 1):** As the warehouse distribution and sorting centres are equipped with sensors, the complete application will be coordinated through advanced warehouse management systems. Each machine will be equipped with sensors to track inventory movements and process orders with a high degree of accuracy. As part of the application, whenever a new order is set to be sorted and delivered, information regarding its location and target will be sent to the principal/primary robot ($R_1$), which in this case is a Fetch and Freight robot (Fig. 5.2), provided by Fetch Robotics [214]. The primary component, Fetch, extends its torso to reach pickup points while a small secondary robot, called Freight, helpfully holds the tote that Fetch will pick items into. Each Fetch robot can have several of these smaller Freight robots supporting the pick-up process. Besides, due to their size they smoothly move around and collect objects throughout the warehouse and hence have been chosen as our principal multi-functioning robot. As a parcel sorting request for a new order is generated, a robot gets the parcel location and plans its path from current location ($P_1$) to go to the given point ($P_2$) for parcel collection. In the context of the application, the primary robot is the centralized decision-maker for the tasks, whereas other tasks provide support when called for.
ii) **Co-packaging Phase (Stage 2):** As the robot is reaching the location \( P_2 \) of the shelves, a mobile piece-picking robot called Magazino [214] is positioned in that area. Magazino, a German start-up company, uses 3D cameras for identifying objects and implements a well-defined grasping technique for collecting objects from the shelves. Thus, the object is picked up and kept in a convenient location. This update is then provided to the principle robot through wireless sensor networks, so that the robot detects the object and pick it up. The next portion of the application involves co-packaging and customization for parcel delivery according to individual needs of the customer. In comparison to more traditional/manual procedures, the robot carries the parcel to the co-packaging centre \( P_3 \) where the well-known robot Baxter, from Rethink Robotics [214] completes the necessary steps of packaging. During these applications, many information processing and analytical tasks are happening in parallel. This is why local or cloud-based offloading may be required for the more efficient performance of the system.

iii) **Package Collection Phase (Stage 3):** After a parcel is customized and co-packaged, it is ready for delivery. At this point, the parcels are put on a conveyer belt to be sent to the collection centre. As updated information is provided to the principal robot \( R_1 \), it moves to the collection point \( P_4 \) to pick up the prepared parcel. While moving, the robot needs to plan its path and communicate with the collection centre to provide an update to the main centre. This creates an opportunity to pass heavy computational tasks to nearby supporting robots for local computation or for assistance with offloading to the cloud.

iv) **Drop-off for Delivery Phase (Stage 4):** As the robot reaches the collection point, it detects the prepared parcel. It uses its own technology to pick up the parcel. Then it updates the main centre and additionally creates an order for the delivery robots (from Starship technologies [215]) to be prepared for the incoming parcel. Then the robot delivers the objects in the drop-off point \( P_5 \) for collection by delivery robot.
As seen from the application details, the principal robot is required to visit five locations ($P_1$-$P_5$) and perform computation-heavy tasks to complete the action. Also, due to the nature of the application, it is time-constrained, which is why additional support from the cloud and other available robots may improve performance. Therefore, other available robots in this cloud networked multi-robot application may help with the communication and local analytical support. For the purpose of the simulation later in the chapter, we have considered the warehouse environment (from Fig. 5.2) where the principal robot is a Fetch and Freight robot $R_1$ and the two supporting robots are Knightscope ($R_3$) from Knightscope Inc. [216] and Tug Robot ($R_2$) from Aethon [217]. Through joint collaboration of the cloud and multi-robot resources, the parcel distribution and sorting process in a smart warehouse is run autonomously to render the parcel ready for delivery, starting from distribution to the eventual drop-off.

5.4 Joint Optimization Problem Formulation

In this section, we model our system for a cloud networked multi-robot application, which leads to the joint optimization problem formulation of offloading decision-making.

5.4.1 System modelling

Our system modelling for task offloading in a multi-robot application integrates four critical factors in its problem formulation to find the optimal/near optimal decisions.

i) Task Offloading and Taskflow Modelling: As seen in Fig. 5.4a, the 40-node task graph in this chapter is derived from the proposed parcel sorting and distribution application graph (Fig. 5.3). In order to maintain consistency with our formulation in the previous chapter, our 40-node task graph is defined by a direct acyclic graph (DAG) and presented as a tuple $D = (\mathcal{T}, K)$. Here each node is considered as a task and known as $T = \{v_j, \ j = 1: t\}$ and $t = |\mathcal{T}|$. We also assume, $K = \{k_{i,j} = \langle t_i, t_j \rangle\}$ and $k = |K|$, where
implies a set of edges and refers to the communication cost from node $t_i$ to $t_j$. More precisely, the term $t_i$ denotes a task $i$ in the task graph where its execution time is dependent on the computation of $t_i^{th}$ task with input data $d_i$. All the task nodes are indicated by Tasks $t_1 \ldots \ldots t_T$. We assume that the nodes on the same level of the DAG (e.g., Tasks 4, 5 and 6) are independent of each other and limited by the “dependency of precedence”. As a result, a task can start only after all preceding tasks on previous level is completed. Finally, highlighted tasks ($P_1 - P_5$) symbolize location constraints of certain tasks for robot $R_1$ and helps identify the starting point of different stages of the operation.

In implementing our offloading approach, our goal is to find the optimal set of decisions and perform suitable offloading in order to complete the task flow within the provided constraints. Since the cloud-based offloading is dependent on the proper trade-off between robot and cloud VM, the offloading decision in this context points to the proper allocation between all the available heterogeneous resources. In this context, these available resources are represented by the robots ($R_1, R_2, R_3$) and cloud VM.

**ii) Robot-Robot Communication and Local Offloading:** In addition to the cloud-based offloading, robot-robot communication also creates a gateway for offloading in cloud networked multi-robot systems. As each robot communicates with other available robots, they create an ad-hoc cloud, which enables the robots to “locally offload” tasks to other available robotic agents. A key factor here is the network topology. The application is considered from the point of view of the primary robot, where all the other robots are providing support. Since the primary robot is making the centralized decisions, hence the other robots are wirelessly connected to the primary one. Even though the other robots may communicate within each other as well, it will maintain communication about each update with the centralized node (primary robot), which suits the scope of our application.
For simulation purposes, the set $\mathcal{R}$ indicate a group of robots that is part of the application, where $R_r \in \mathcal{R}, \forall \{i = 1: n\}$. Here $R_i$ denotes the selected robot and $n$ indicates the total number of robots available that may communicate with each other. Depending on the type of decision-making, the robots in this context locally offload tasks in two scenarios, a) to offload to another robot for computation support, b) to offload to another robot that is used as a hub for further offloading to the cloud VM. In either situation, the available robots support the principle robot to reduce workload and improve performance.

For local (robot-robot) communication though, we present a popular/well-established communication model \[218\], where the communication parameters (energy/time) are considered based on the distance among the robots. For a threshold distance of $l_0$, if the distance ($l'$) between two robots is less than $l_0$, then the “free space” channel model is considered. Hence the communication energy and time is derived as:

$$E_{LO}(t_i) = (e_{base} + \varepsilon_{fs}.l'^2) \times d(t_i): l' < l_0$$

(5.1)

$$T_{LO}(t_i) = (e_{base} + \varepsilon_{fs}.l'^2) \times \frac{d(t_i)}{P_{LO}}: l' < l_0$$

(5.2)

Here $e_{base}$ is the baseline energy consumption for operating the transmitter radio for local communication. As mentioned, for $l' < l_0$ the transmission energy consumption is assumed to be a “free space” channel model and hence presented by $\varepsilon_{fs}.l'^2$. $P_{LO}$ is the processing power of robot ($R_i$) local offloading, whereas $d(t_i)$ is the information being offloaded. However, when distance ($l'$) is greater than threshold, it considers “multipath fading” channel model for communication and the transmission energy consumption is $\varepsilon_{mf}.l'^4$. Consequently, the complete energy and time calculation for local offloading is:

$$E_{LO}(t_i) = (e_{base} + \varepsilon_{mf}.l'^4) \times d(t_i): l' \geq l_0$$

(5.3)

$$T_{LO}(t_i) = (e_{base} + \varepsilon_{mf}.l'^4) \times \frac{d(t_i)}{P_{LO}(R_i)}: l' \geq l_0$$

(5.4)
The communication parameters and the resulting calculation for local offloading are performed based on the locations of the robots with respect to each other and therefore are subjected to the decision-making.

More details about the parameters are provided later in the chapter.

iii) **Workspace and Path Planning:** Our application workspace in Fig. 5.4c is derived from Fig. 5.2, which depicts a warehouse environment. The warehouse workspace is represented by a grid, sized $m \times n$. Each cell in the grid points to a uniform cell location. As the robots move through each cell, they eventually map out a path plan for robot movement. In the context of our work, the grid-based model is chosen as our workspace because of its ease with calculation of distances, representation of obstacles and scalability with respect to condition changes. As mentioned previously in section 4.4.2, each cell is denoted by $L = (X, Y)$, where $X = \{x = 1: m\}, Y = \{y = 1: n\}$ and $\forall L \in [1: l]$, where total number of cells $l = 36$. Each cell $L$ is a location the robot chooses for a given task. The whole workspace is characterized by orderly numbered grids and size of the grids determines how many cells there are.

Certain cells are considered as obstacles and are off-limits to all the robots. These obstacles are directly adopted from the warehouse environment (Fig. 5.3) and presented here as $O_l$. The application also considers the task constraints of certain robots, meaning selected tasks are allocated to pre-defined robots as well as fixed locations (e.g., getting a parcel from a fixed warehouse shelf). These tasks are presented as $T_{i,l} = \langle i, l \rangle$; where task $i$ is constrained to location $l$. The robots are aware of these constraints at the time of operation and hence move accordingly. Finally, the application taskflow (Fig. 5.4a) is integrated with the workspace in the form of starting point $P_1$ and finish point $P_4$. This
helps the robot to identify the task constraints with respect to locations as well as get a clear idea of the task sequence.

One of the key parameters for offloading decision-making is the choice of location for each task. In this context, the relative parameter that helps identify these decisions is the distance value between cells that indicates the total distance the robot covers. As movement also results in added energy, less amount of movement (unless necessary) is a priority for the robot in order to save energy. Hence the accurate distance between cells in this grid workspace is calculated using a modified A-star method [205]. The details of the process have already been presented in section 4.4.2. By using the A-star method, we calculate the distance \( f(l)_{a,b} \) between any two points \((x_1, y_1)\) and \((x_2, y_2)\). Eventually it is used to gain the total distance coverage for each robot’s movement \( D_{total} \) where the tasks are part of the movement set \( m(t) \).

\[
D_{total} (R_r) = \sum_{t_i \in m(t)} f(l)_{a,b}
\]  

(5.5)

Using these cost values, we find movement energy and time, which is then used during the calculation of optimal offloading and movement decisions. More details of these calculations are presented during the formulation.

iv) **Access Point Selection and Bandwidth Estimation:** As already highlighted, one of the key benefits of CNR is the on-demand movement capability of the robot that can be utilized to gain better bandwidth. Depending on the choice of access point, the robot may achieve its desired bandwidth; this plays an important role in task offloading and system performance. AP selection is broadly classified into two categories. The first category is “online AP selection” where the choice of AP is made during the on-line phase, based on relatively stable features and selection criteria conveying physical or statistical meanings. However, this tends to consume high levels of energy as well as perform poorly for
complex scenarios such as ours. In comparison, the second category of “offline AP selection” is more suitable for our work. Here all APs are defined by their score function and availability. Based on these, the most suitable solution is selected as the choice of AP. For an unstructured problem set like ours, the offline approach suits a GA-based scheme by reducing the level of complexity and compensating for the dynamic nature of the rest of the implementation of our algorithm.

In order to implement the latter approach, the workspace is set up with several APs in different locations for connecting to the cloud via the Internet. However, the robot needs to share the WiFi network with other regular users. Depending on the path planning and its eventual location, the robot will have the option of connecting to one of multiple APs. As mentioned, here we implement the offline approach and provide a communication model to setup our workspace with variable numbers of APs and users. Depending on the choice of AP, the robot accesses different bandwidths at different locations. Hence the choice of AP becomes a priority. As the WiFi network is shared with other users, each AP has certain users associated with it. Hence, the robot has to estimate the “fair-share bandwidth” of a given location for communication requirements. Given the size of the application workspace, a complete area is expected to be covered by at least one AP and hence robots are always within the coverage of Internet connectivity. Moreover, the number of users associated with each AP for the duration of the application is considered fixed, since the time period is relatively small and hence less sensitive to dynamic changes.

Following up from our communication model in section 4.4.3, the AP is defined by \( \alpha \) and the total sets of AP are presented as \( \mathcal{A} \). The bit rate function is represented by \( b (\cdot, \cdot) \) as \( b : (\mathcal{L} \times \mathcal{A}) \to B \). Here \( r(l, \alpha) \in B \) is the bit rate at which each robot can individually transmit data from location \( l \in \mathcal{L} \) with AP \( \alpha \in \mathcal{A} \), where \( \mathcal{A} = \{1 \ldots \alpha\} \). We consider \( B \)
to be the set of bit rates available with the technology being used. Given this set, the individual bit rate $r$ will depend on the location and the AP. The further the robot is from a given AP, the lower the value of $r$ will be. This bit rate value is known to the robot for any given location during the task operation. Since each access point is shared by a number of users, the bandwidth at any given location is shared among the number of users as well. So the robot’s ($R_r$) throughput/bandwidth $\beta$ at location $l$ can be estimated according to protocol IEEE 802.11 WLANs [206], which has already been presented in

Table 5.1: Additional notations (following on from Table 3.1 and Table 4.2)

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{R_r}$</td>
<td>Energy consumption of robot $R_r$</td>
</tr>
<tr>
<td>$E_{\text{limit}}(R_r)$</td>
<td>Energy limit of robot $R_r$</td>
</tr>
<tr>
<td>$D_{\text{total}}(R_r)$</td>
<td>Total distance covered by the robot $R_r$</td>
</tr>
<tr>
<td>$P_i(R_r)$</td>
<td>Robot $R_r$ processing power for sending instruction to cloud</td>
</tr>
<tr>
<td>$P_u(R_r)$</td>
<td>Robot $R_r$ processing power for uploading data to cloud</td>
</tr>
<tr>
<td>$P_r(R_r)$</td>
<td>Robot $R_r$ processing power for on-board computation</td>
</tr>
<tr>
<td>$P_{cc}(R_r)$</td>
<td>Robot $R_r$ processing power during cloud computation</td>
</tr>
<tr>
<td>$P_{\text{mov}}(R_r)$</td>
<td>Robot $R_r$ processing power during robot movement</td>
</tr>
<tr>
<td>$P_d(R_r)$</td>
<td>Robot $R_r$ processing power for WSN communication</td>
</tr>
<tr>
<td>$P_{LO}(R_r)$</td>
<td>Robot $R_r$ processing power for local offloading</td>
</tr>
<tr>
<td>$v(R_r)$</td>
<td>Robot $R_r$ movement velocity</td>
</tr>
<tr>
<td>$H_r(R_r)$</td>
<td>Robot $R_r$ transfer rate for WSN communication</td>
</tr>
<tr>
<td>$S_{R_r}$</td>
<td>Clock speed of robot $R_r$ processor</td>
</tr>
<tr>
<td>$BPI(R_r)$</td>
<td>Bits per instruction for robot $R_r$</td>
</tr>
<tr>
<td>$CPI(R_r)$</td>
<td>Average number of clock cycles per instruction for robot $R_r$</td>
</tr>
</tbody>
</table>
equation 4.3 (previous chapter). Assuming each AP is shared by multiple users and all WiFi users use the same packet size, the resulting throughput \( \beta(l, \alpha) \) is the “fair-share” bandwidth the robot receives at location \( l \), if it selects AP \( \alpha \). The throughput equation in (4.3) is presented in more detail to improve accuracy, but only the necessary information has been provided since it captures essential features of the packet scheduling for 802.11 MAC in the present form. Based on the information provided, the AP selection is integrated with the robot’s path plan, local R-R offloading and R-C offloading, so that it gains access to better bandwidth for faster communication with the cloud.

5.4.2 Cost functions

Our system modelling for task offloading in a multi-robot application integrates four critical factors in the problem formulation. Two types of factors are considered for the calculation of energy and latency cost functions, i) fixed parameters—task input, robot and cloud VM processing power; and ii) variable parameters—local offload, cloud-based offloading, bandwidth and movement. Table 3.1 and Table 4.2 have already listed the basic notations for the calculation of energy and latency cost functions, based on task assignments. In addition, Table 5.1 presents the additional notation necessary for this problem formulation. Our goal is to identify optimal offloading decisions with collective consideration of mobility and communication in a cloud networked multi-robot application that results in minimum consumption of the robotic energy.

i) Robotic Energy Calculation: The total energy \( E_{\text{total}} \) consists of energy from all tasks partitioned into on-board, local offload and cloud allocation, as seen in (5.6). Even though, primary robot is the decision-maker and provides majority of the actuation, the following equation provides total energy for all of the tasks (on-board, on cloud and communication) involved as part of the application, which is completed by all the robots together.
\[ E_{total} = \sum_{i=1}^{\text{|𝒯|}} I_{t_i} \cdot L_{t_i} \cdot E_R(t_i, r, l) + \sum_{i=1}^{\text{|𝒯|}} -I_{t_i} \cdot R_{t_i} \cdot L_{t_i} \cdot A_{t_i} \cdot E_C(t_i, r, l, \alpha) \]  (5.6)

Similar to previous formulations, \( I_{t_i} \) denotes the offloading decisions and \( \neg \) is the NOT operator, signifying the tasks that are offloaded to the cloud. \( L_{t_i} \) is the unknown variable that indicates the location for each task, whereas \( A_{t_i} \) indicates the selection of AP for offloaded tasks. Finally, the term \( R_{t_i} \) refers to the selection of the robot for offloaded tasks, which signifies whether the task is offloaded to the cloud from on-board a given robot or the task is transferred to a nearby robot for further offloading to the cloud facilities.

The total energy is the summation of all the tasks that are divided into two parts. Tasks performed on-board are denoted by \( E_R(t_i, r, l) \), which indicates they are dependent on the selection of task, robot (local offload) and location. The tasks that are taking place on the cloud VM are identified by \( E_C(t_i, r, l, \alpha) \), which means it is additionally dependent on the selection of AP for cloud-based offloading. Equation (5.6) is further elaborated as seen in equation (5.7) and (5.8):

\[
\begin{align*}
E_R(t_i) & = \frac{E_{MOV}}{\text{Movement}} + \frac{E_{WSN}}{\text{Wireless Sensor}} + \frac{E_{LO}}{\text{Local offload}} + \frac{E_{RC}}{\text{Local Computation}} \quad (5.7) \\
E_C(t_i) & = \frac{E_{MOV}}{\text{Movement}} + \frac{E_{WSN}}{\text{Wireless Sensor}} + \frac{E_U}{\text{Data Upload}} + \frac{E_I}{\text{Sending Instruction}} + \frac{E_{CC}}{\text{Cloud Computation}} \quad (5.8)
\end{align*}
\]

Depending on the selections of \( I_{t_i}, R_{t_i}, L_{t_i} \) and \( A_{t_i} \), we calculate the robot’s energy and task completion time. For tasks taking place on the robot, the parameters include the movement energy \( E_{MOV}(t_i, l) \), data collection \( E_{WSN}(t_i, l) \), computation energy \( E_{RC}(t_i) \) and local offloading energy \( E_{LO}(t_i) \). As seen from the components of the equation, both \( E_{LO}(t_i) \) and \( E_{RC}(t_i) \) is dependent on location and the choice of robot.
\[
E_{LO}(t_i) = \begin{cases} (e_{base} + \varepsilon_{fs} l_i^2) \times d(t_i): l' < l_0 \\ (e_{base} + \varepsilon_{mf} l_i^4) \times d(t_i): l' \geq l_0 \end{cases}
\]

\[
E_{RC}(t_i) = P_r(R_r) \times CPI(R_r) \times \frac{N(t_i)}{S_{R_r}}
\]

As previously stated, \( E_{LO}(t_i) \) points to the energy consumed for tasks being offloaded locally. Depending on the distance of robots from each other, the process either follows a “free channel” model (when \( l' < l_0 \)) or “multipath fading” (when \( l' > l_0 \)). The on-board computation energy is denoted by \( E_{RC} \), which is dependant on the task and is calculated for the robot that performs it.

The rest of the parameters is calculated in the same manner as explained in chapters 3 and 4 by cumulating values for each robot \( R_r \). These equations are presented as follows:

\[
E_l(t_i) = P_l(R_r) \times BPI(R_r) \times \frac{N(t_i)}{\beta(l,a)}
\]

\[
E_u(t_i) = P_u(R_r) \times \frac{d(t_i)}{\beta(l,a)}
\]

\[
E_{CC}(t_i) = P_{cc}(R_r) \times CPI(R_r) \times \frac{N(t_i)}{S_c}
\]

\[
E_{Mov}(t_i, l) = \sum_{t_i \in m(t)} P_{mov}(R_r) \times \frac{l_{ab}}{v(R_r)}
\]

\[
E_{WSN}(t_i) = \sum_{t_i \in v(t)} P_d(R_r) \times \frac{d(t_i)}{\mathcal{H}_r(R_r)}
\]

Using equations (5.11-5.15), we calculate values for the above-mentioned parameters based on our offloading decision-set. Specifically, in equation 5.14, \( E_{Mov}(t_i, l) \) indicates the movement energy to go to a particular location for offloading, where a key parameter is robot velocity \( v(R_r) \) that is different for each robot. Based on this, total movement energy/time for each robot is calculated in addition to total distance \( D_{total} \). On the other hand, energy \( E_{WSN}(t_i, l) \) is describes energy consumption of robot during data collection.
from WSN. Given the scale of the operation, the WSN communication power \( P_d(R_r) \) and data transfer rate \( \mathcal{H}_r(R_r) \) for each robot is equal.

\textbf{ii) Time Calculation:} Time calculation is carried out in a similar way to that of the robot energy calculation where tasks are divided between on-board and cloud-based allocations. The on-board tasks also include the locally offloaded ones for further processing. The time calculation uses the exact same communication model for offloading, as mentioned in the previous section for energy consumption. All the equations are presented below:

\[
\mathcal{T}_{total} = \sum_{i=1}^{[\mathcal{T}]} I_{t_i} \cdot L_{t_i} \cdot \mathcal{T}_R(t_i, r, l) + \sum_{i=1}^{[\mathcal{T}]} I_{t_i} \cdot R_{t_i} \cdot L_{t_i} \cdot \mathcal{A}_{t_i} \cdot \mathcal{T}_C(t_i, r, l, \alpha) \tag{5.16}
\]

\[
T_R(t_i) = \sum_{t_i \in \mathcal{M}(t)} \frac{l_{a,b}}{v(R_r)} + \sum_{t_i \in \mathcal{V}(t)} \frac{d_d(t_i)}{\mathcal{H}_r(R_r)} + \begin{cases}
(e_{base} + e_{f_s} \cdot l'^2) \times \frac{d(t_i)}{P_{LO}(R_r)} : l' < l_0 \\
(e_{base} + e_{m_f} \cdot l'^4) \times \frac{d(t_i)}{P_{LO}(R_r)} : l' \geq l_0
\end{cases} \tag{5.17}
\]

\[
T_C(t_i) = \sum_{t_i \in \mathcal{M}(t)} \frac{l_{a,b}}{v(R_r)} + \sum_{t_i \in \mathcal{V}(t)} \frac{d_d(t_i)}{\mathcal{H}_r(R_r)} + \frac{d(t_i)}{\beta(l, \alpha)} + \frac{N(t_i)}{\beta(l, \alpha)} \tag{5.18}
\]

Here the processor speed for cloud-based computation is \( S_c \), which is much larger than the processing speed \( S_{R_r} \) of each robot. Thus, the cost functions for the robot and the cloud VM task completion time is used to calculate the overall results for \( \mathcal{T}_{total} \).

\textbf{iii) Joint Optimization Problem:} This chapter addresses a four-fold problem. Based on the problem formulation for the proposed application, the objective is for the robot to find
the optimal decisions for cloud-based offloading, local offloading, path planning and AP selection, together within the constraints imposed, thus providing task offloading for a cloud networked multi-robot system with collective consideration for motion and connectivity in decision-making. Let the following variables indicate their respective decisions for each task.

- $I_{t_i}$ = Offloading decision for each task. Here $I_{t_i}$ indicates that the task is executed on robot $R_r \in \mathcal{R}, \forall \{r = 1:n\}$. And $\neg I_{t_i}(0)$ specifies that task $t_i$ is offloaded to cloud VM. For our formulation, the total number of robots $n=3$. So, possible task allocations decisions on-board of robot are $R_1, R_2, R_3$.

- $R_{t_i}$ = Selection of robot $R_r$ for offloading a task $t_i$ to the cloud, where $\mathcal{R}_{t_i} \in I_{t_i} = 0$. This decision is for identifying which robot will offload the task to the cloud.

- $L_{t_i}$ = Location for each task where the set consists of total $l$ values ($L = 1…l$). For our formulation, $l = 36$.

- $\mathcal{A}_{t_i}$ = Selected AP for the offloaded task, where AP set has total $\alpha$ values ($\mathcal{A} = 1…\alpha$). In our problem, $\alpha = 4$.

Based on the proposed application scenario and the problem formulation, the objective is to minimize the total energy ($E_{total}$) consumption of robots in order to meet the time constraint ($T_{Deadline}$) and individual energy constraint ($E_{R_{lim}}$) of each robot ($E_{R_r}$).

**Find:** \{ $I_{t_i}$, $\{R_{t_i}\}$, $\{L_{t_i}\}$, $\{\mathcal{A}_{t_i}\}$, $\forall \mathcal{T} = \{v_j, j = 1:t\}$, and $t = |\mathcal{T}|$ \} to minimize: $E_{total}$

s.t.: $T_{total} \leq T_{Deadline}$ and $E_{R_r} \leq E_{limit}(R_r)$
5.5 GA-based 4-Layer Decision-Making Scheme

As previously mentioned, Genetic algorithm (GA) is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection that represents an intelligent exploitation of random searches in order to determine optimal solutions. Traditionally, GA is widely used in various fields due to its global acceptability and high efficiency as well as its impressive stability [65]. The heuristic approach of GA works efficiently for large-scale multi-objective optimization problems, because it approximates brute force without enumerating all the elements, thereby bypassing performance issues specific to an exhaustive search [207], which suits the NP-complete problem set in this work. Given our scenario, GA is a suitable approach to identify optimal offloading decisions in cloud networked multi-robot systems.

For Multi-Robot Task Allocation (MRTA) optimization problems, the notable current studies centre on various algorithms (e.g., heuristic [219], timed automata model [220], market-based approach, swarm intelligence [154], task-grouped improved static allocation algorithm, decentralized approach [183] etc.) that perform successfully in solving optimization problems. However, all these approaches mostly consider a single variable (task/robot/path planning/ allocation) to tackle the problem. In order to keep up with the rapid increase in technology and handle more complex systems, the progressive approach is to prepare more rigid and comprehensive techniques with interdependent parameters. This motivates our work where we have considered four variables (task offloading, robot selection for offloading, path selection and access point selection) as part of a multi-layer decision-making set. The objective is to solve the optimization problem for a multi-robot system and identify these key decisions in order to minimize $E_{total}$. This joint optimization exploits the benefits from each of the individual elements and considers all four symbiotic parameters jointly as part of the solution and hence
simultaneously explores the search space for optimized and cost-efficient results, which are reflective of system performance. For our modified GA-based scheme in the multi-robot context, the following steps need to be taken (as shown in Fig. 5.5).

Figure 5.5: Steps of GA-based multi-layer decision-making scheme

Figure 5.6: Modified chromosome encoding for GA-based 4-layer decision-making scheme for cloud networked multi-robot system
Novel Four-layer Chromosome Encoding: The genetic algorithm (GA) scheme initiates by randomly generating a primary population (P), consisting of individuals whose genetic material represents sample points in the solution space. As seen in Fig. 5.6, these individuals are a collective unit of novel 4-layer solutions known as chromosomes. Here each layer points to a separate set of decisions. These decisions are selected from the possible options in the search space and are presented in the following order: i) offloading decisions, ii) robot selection for offloaded tasks, iii) location selection for each task, and iv) selected AP for the offloaded tasks. Therefore, 

\[ P = \begin{bmatrix}
I_{t_1} & I_{t_2} & \ldots & I_{t_i} & \ldots & I_{t_T} \\
R_{t_1} & R_{t_2} & \ldots & R_{t_i} & \ldots & R_{t_T} \\
L_{t_1} & L_{t_2} & \ldots & L_{t_i} & \ldots & L_{t_T} \\
A_{t_1} & A_{t_2} & \ldots & A_{t_i} & \ldots & A_{t_T}
\end{bmatrix} \]

An example of the decision set is given in Fig. 5.6 as: \( I_1 = [1] [\times] [13] [\times] \), which suggests task 1 is completed on robot \( R_1 \) at cell 13 in the workspace. As a result, no offloading is required by any robot, hence no AP is selected. Another example is: \( I_2 = [0] [3] [8] [2] \). This indicates that the task is completed on cloud VM (0), while it is offloaded to the cloud by robot \( R_3 \) from workspace cell 8 through AP 3.

In addition, tasks are further divided into two groups. The tasks that are constrained to any fixed location/allocation (constrained to robot), are considered as un-offloadable, whereas the rest of the tasks are offloadable. The un-offloadable tasks, their fixed allocations and locations are indicated in Fig. 5.3. In order to consider the un-offloadable tasks, the encoded chromosome in this section is further modified by fixing constrained tasks to their fixed location (\( L_{t_i} \)) and allocation (\( I_{t_i} = 1 \) for robot \( R_1 \)), as seen in Fig. 5.6. In terms of the GA scheme, this would save unnecessary latency during the latter phase by not allowing these particular bits to be changed. Hence these solutions remain fixed.

It also provides a real-world context where certain tasks (e.g., data collection, parcel pickup, delivery etc.) tend to be in fixed locations or fixed to certain robots. Such constraints may be added in the scheme to compensate for those scenarios.
**ii) Fitness (Parameter) Calculation and Evaluation:** In this stage, each individual/solution in the population is evaluated by invoking the fitness function $f$ to measure the quality of the solution in the search space. For the given problem, the fitness is considered as $f = E_{total}$. Since the objective is to minimize energy, the term “lower energy” indicates “higher fitness measure” in this context. Similarly, “best fitness measure” also designates the solution that results in the “lowest value”. In order to determine fitness value, the calculation of critical performance parameters i.e., energy ($E_{total}$), time/delay ($T_{total}$), distance ($D_{total}$), individual energy ($E_{R_r}$) and individual robot distance coverage ($D_{R_r}$) are performed using equations (5.1)–(5.18). As taskflow is divided into several levels, a breadth-fast search is performed to identify task dependencies and divide them into groups for level-wise calculation. A pseudo-code in Fig. 5.7 explains step-by-step calculation for fitness measure and other parameters.

- *Calculation of $E_{total}$ (Fitness) & $E_{R_r}*$

  Similar to our previous approach in chapter 4, a level-wise calculation is performed to obtain the values of robot energy for tasks offloaded to the cloud $E_c(t_i, l)$ and tasks completed on the robot $E_R(t_i, l)$. Since energy values are additive, the resulting energy $E_{total}$ from the corresponding level is collected and added to the total values. Thus, energy from all the levels adds up to ultimately determine the final updated value of total energy $E_{total}$. Another important parameter is the energy consumption of each robot $E_{R_r}$. Based on the selection of the robot for local task completion or local offloading, energy is consumed for tasks as seen in the calculation. For each consumption, the energy cost corresponding to a task is added to the selected robot $R_r$. In this way, for each task, the energy values is added to corresponding robots (based on task allocation) and total energy consumption by each robot $E_{R_r}$ is calculated.
Input: Workspace, DAG, $I_t$, $R_t$, $L_t$, $A_t$

Output: $E_{total}$, $T_{total}$, $D_{total}$, $E_{R_r}$, $D_{R_r}$

1: Initialize $E_{total}$, $T_{total}$, $D_{total}$, $E_{R_r}$, $D_{R_r}$

2: for each level $H_j \in H_{total}$ do /* Calculate energy, time, distance*/

3: for each task $t_i \in H_j$ do

4: $H_j(I_t) \in \{0, R_r\}, \forall \ r = \{1, ..., n\}$ /*Find Task Allocation */

5: $H_j(R_t) \in R_r, \forall \ r = \{1, ..., n\}$ /*Find Offloading Robot */

6: $H_j(L_t) \in L = \{1, ..., l\}$ /*Find Task Location */

7: $H_j(A_t) \in A = \{1, ..., a\}$ /*Find Access Point */

8: if $I_t(R_r)$ (task on robot $R_r$) do

9: for $R_r \in R, \forall \ r = \{1, ..., n\}$ do

10: if $t_i \in m(t)$ [movement task set] do

11: $E_R(t_i, l) = E_{Mov}(t_i, l, r) + E_{WSN}(t_i, l, r) + E_{LO}(t_i, l, r) + E_{RC}(t_i, l, r)$

12: $E_{total} = E_{total} + E_R(t_i, l, r)$

13: $T_R(t_i) = T_{Mov}(t_i, l) + T_{WSN}(t_i, l, r) + T_{LO}(t_i, l, r) + T_{RC}(t_i, l, r)$

14: $D_{total} = D_{total}(R_r) + f(l)_a,b$ /*Total Distance*/

15: $D_{R_r} = D_{R_r} + D_{total}(R_r)$ /*Individual Robot Distance*/

16: else do

17: $E_R(t_i, l) = E_{WSN}(t_i, l, r) + E_{LO}(t_i, l, r) + E_{RC}(t_i, l, r)$

18: $E_{total} = E_{total} + E_R(t_i, l, r)$

19: $T_R(t_i) = T_{WSN}(t_i, l, r) + T_{LO}(t_i, l, r) + T_{RC}(t_i, l, r)$

20: end if

21: $E_{R_r} = E_{R_r} + E_{R_r}(t_i)$ /*Calculate individual robot energy*/

22: end for
else \(\neg I_i(0) \) (task on cloud) do

if \(t_i \in m(t) \) [movement task set] do

\[
E_C(t_i, l, r, \alpha) = E_{Mov}(t_i, l, r, \alpha) + E_{WSN}(t_i, l, r, \alpha) + E_U(t_i, l, r, \alpha) + E_I(t_i, l, r, \alpha) + E_{cc}(t_i, l, r, \alpha)
\]

\[E_{R_r}(t_i) = E_C(t_i, l, r, \alpha) \quad /\text{Calculate individual robot energy} /
\]

\[E_{total} = E_{total} + E_C(t_i, l, r, \alpha)
\]

\[T_C(t_i, l, r, \alpha) = T_{Mov}(t_i, l, r, \alpha) + T_{WSN}(t_i, l, r, \alpha) + T_U(t_i, l, r, \alpha) + T_I(t_i, l, r, \alpha) + T_{cc}(t_i, l, r, \alpha)
\]

\[D_{total} = D_{total}(R_r) + f(l)_{a,b} \quad /\text{Total Distance} /
\]

\[D_{R_r} = D_{R_r} + D_{total}(R_r) \quad /\text{Individual Robot Distance} /
\]

else do

\[E(t_i) = E_{WSN}(t_i, l, r, \alpha) + E_U(t_i, l, r, \alpha) + E_I(t_i, l, r, \alpha) + E_{cc}(t_i, l, r, \alpha)
\]

\[E_{R_r}(t_i) = E_C(t_i, l, r, \alpha)
\]

\[E_{total} = E_{total} + E_C(t_i, l, r, \alpha)
\]

\[T_C(t_i, l, r, \alpha) = T_{WSN}(t_i, l, r, \alpha) + T_U(t_i, l, r, \alpha) + T_I(t_i, l, r, \alpha) + T_{cc}(t_i, l, r, \alpha)
\]

end if

\[E_{R_r} = E_{R_r} + E_{R_r}(t_i) \quad /\text{Calculate individual robot energy} /
\]

end if

end for

if \(T_R(t_i, l, r) > T_C(t_i, l, r, \alpha) \) do

\[T_{total} = T_{total} + T_R(t_i, l)
\]

else do

\[T_{total} = T_{total} + T_C(t_i, l, \alpha)
\]

end for

Calculate \(E_{total}, T_{total}, D_{total}, E_{R_r}, D_{R_r}\)

Figure 5.7: Pseudo-code for robotic energy, time and distance calculation
• Calculation of $\mathcal{T}_{\text{total}}$

Since the tasks on the same level ensue in parallel, total time for tasks is not directly additive. Therefore, a slight modification is required to compensate for parallel tasks. Tasks from each level are further divided into two types based on allocations (0 or $R_r$) and put into two different lists. Total time for tasks on the cloud and on the robots is calculated separately. A simple comparison is then performed between the cumulative time values (for each level) to resolve which takes longer. Here the higher value between the two is considered as the actual latency/time cost from that DAG level. Similar to the process previously used, the values from each level is then calculated and added altogether in order gain the results of the total task completion time $\mathcal{T}_{\text{total}}$ for the application.

• Calculation of $D_{\text{total}}$ & $D_{R_r}$

Distance is calculated according to the fourth layer of the chromosome, which equates to the location decisions ($L_{t_i}$). For every task that is part of set $m(t)$, distance cost between the corresponding points for each robot is calculated (via the A-star method), as seen in chapter 4. Based on the allocation (robot or cloud), the distance value is calculated sequentially for each level and distance values are added for either the cloud or robot-based allocation $D_{\text{total}}(R_r)$. These values are then added to overall values to ultimately gain the total distance covered by the robots $D_{\text{total}}$. In addition, for each distance cost, the values are also added to the corresponding individual robots ($R_r$). In this way, the individual distance coverage $D_{R_r}$ for each of the robots is calculated.

iii) Selection Phase (Parameter) Calculation and Evaluation: After the fitness calculation stage, the mating pool is filled in iteratively from the current generation. From then on, two chromosomes are randomly selected in each pass and the individual with the
higher fitness measure (i.e., low energy) is finalized to fill the mating pool. The process is repeated until the mating pool is completely filled in. We also adopted elitism by keeping the historical best solution in the mating pool. Thus, the next generation is produced by selecting individuals with higher fitness measures through the selection probabilities to produce offspring via genetic operators.

Moreover, the infeasible solutions are also dealt with during the selection stage wherein these solutions are given very low fitness measures, which eventually results in high energy values. Therefore, these solutions are never picked up for the mating pool (since their fitness will be always inferior to the feasible answers) and the solution will remain accurate.

**iv) Crossover Phase:** The strategy employed by crossover is to construct new individuals from existing high-performance individuals by recombining subcomponents (Fig. 5.8). In this scenario, two selected chromosomes from the previous phase “reproduce” in crossover section and produce “offsprings”. For this phase, the “uniform crossover” process is considered as it uses a fixed mixing ratio between two parents. This process allows parent chromosomes to contribute in the gene level rather than the segment level. Similar to our previous approaches (chapters 3, 4), the individual bits in the string are compared between their two parents during this stage, and all the bits are swapped with a fixed probability. As mentioned, infeasible solutions are removed by giving them low fitness measures in order to avoid possible degradation of GA performance. However, if any such solution is still produced at crossover, it would be eliminated at the fitness calculation or mating pool generation stage of the following pass of genetic algorithm.
<table>
<thead>
<tr>
<th>Layer</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5.8: Crossover phase for GA-based offloading scheme
v) **Mutation Phase**: At the end of the selection and crossover phases, the mutation phase is performed on the population that contains possible new solutions. However, the chromosomes may become too similar to each other in some cases. At this point, the mutation operator updates these individuals by independently modifying one or more of the gene values of an existing individual. More specifically, a portion of new individuals have some of their bits flipped with low probability (0.5) by the operator. This is done
intentionally, so as to ensure proper diversity and the possibility of finding the global optimum in the search-space. However, contrary to previous chapters, the solution here has four layers. So, the mutation process is finalised separately on each of the layers of the solutions (Fig. 5.9). Similar to the “crossover” section, the infeasible results are given low fitness measures so as to avoid the quality of results becoming poorer. The constrained elements (fixed allocation and location of task) in the chromosome are also dealt with in this section. For this purpose, the chromosome is further modified at the end of this stage to compensate for the constrained tasks by forcefully changing the elements to their fixed allocation and location. After this, we obtain a new population of individuals and the process is continued in the same way.

vi) Self-Stopping Criteria: A self-stopping criteria is embedded so that the process doesn’t evoke unnecessary latency or processing power. Since our GA scheme provides the “lowest energy” (best fitness measure) after each generation, it will only stop when there is no change in the best fitness measure for a prefixed (determined by user) number of generations. At that point, GA is terminated immediately, and the result is considered as near-optimal.

5.6 Simulation Results and Analysis

We ran extensive simulations and analysed the different aspects of our GA scheme for the multi-robot with cloud (GAMRC) approach; these are: decision-making, fitness measure, offloading, AP usage, robot’s usage and path planning. Based on these, we assess the quality of the algorithm and its effectiveness in the context of the application. We further determine the performance of our scheme by comparing our findings with a GA scheme for a single cloud-aided robot (GASRC) from chapter 4 where mobility-driven and communication-aware task offloading was performed in similar applications.
[189], but only for a single robot application. For such single robot approaches, all tasks are allocated between the robot and the cloud VM. The mobility indicates the location selection for each task, whereas the AP selection points to the gateway towards cloud communication for allocated tasks. The comparison between the single robot and the multi-robot cloud-based approach in this section would help identify/verify the benefits of the multi-robot framework in task offloading. More precisely, this would point out how additional robots could help out in the local computation sharing of tasks as well as in offloading to the cloud.

In addition, we also compare the results with a GA scheme for a multi-robot (GAMRB) on-board approach where only R-R communication has been used for task completion. This approach doesn’t consider cloud infrastructure as a possible source of allocation. Through this comparison of results, the impact of the cloud in such applications may be recognized. Both benchmark approaches (GASRC and GAMRB) are calculated via a genetic algorithm-based method as well. Hence the results from these methods are near-optimal. As for authentication of these benchmarks, the results have been previously evaluated properly via comparison with exhaustive search [183], All-on-Robot (AoR) approach [189], greedy algorithm [182] as well as a single robot offloading method with fixed movement and bandwidth [183], as seen in chapters 3 and 4. Therefore, comparison of our proposed scheme (GAMRC) in this chapter with such credible reference methods helps to validate our findings.

Table 5.2 describes the parameters required for setting up the simulation. The 36-cell workspace considered in this work is inspired from the industrial warehouse presented in Fig. 5.2. The obstacle cells are marked black (Fig. 5.4c) and point to cells that are off limits for movement selection. The details of the obstacles are presented in the table. The 40-node taskflow defines tasks that need to be completed in the constrained scenario,
Table 5.2: Simulation parameter setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (Min: $E_{total}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Nodes</td>
<td>40</td>
</tr>
<tr>
<td>Deadline ($T_{Deadline}$)</td>
<td>Time (150 sec)</td>
</tr>
<tr>
<td>Population</td>
<td>500</td>
</tr>
<tr>
<td>Stopping</td>
<td>1500 generations without change in fitness measure</td>
</tr>
<tr>
<td>Obstacle Cells</td>
<td>Cells 9, 18, 20, 33, 36</td>
</tr>
<tr>
<td>AP &amp; Users</td>
<td>4 APs and each AP is associated with 3 users</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Robot No.</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{R_r}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_r(R_r)$</td>
<td>55 W</td>
<td>40 W</td>
<td>25 W</td>
</tr>
<tr>
<td>$P_u(R_r)$</td>
<td>80 W</td>
<td>60 W</td>
<td>40 W</td>
</tr>
<tr>
<td>$P_{cc}(R_r)$</td>
<td>20 W</td>
<td>15 W</td>
<td>10 W</td>
</tr>
<tr>
<td>$P_{LO}(R_r)$</td>
<td>11 W</td>
<td>12 W</td>
<td>13 W</td>
</tr>
<tr>
<td>$P_m(R_r)$</td>
<td>50 W</td>
<td>35 W</td>
<td>20 W</td>
</tr>
<tr>
<td>$CPI(R_r)$</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>$BPI(R_r)$</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation and Location Constraint</th>
<th>Task</th>
<th>Zone</th>
<th>Task</th>
<th>Zone</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>25, 26</td>
<td>5</td>
<td>Robot $R_1$</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>31</td>
<td>27</td>
<td>5</td>
<td>Robot $R_1$</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>35</td>
<td>37</td>
<td>24</td>
<td>Robot $R_1$</td>
</tr>
</tbody>
</table>
where selected tasks \((P_1 - P_5)\) must be completed by robot \(R_1\). Hence the location constraints for these tasks are also presented in the table. The time deadline \((T_{\text{deadline}})\) for the proposed application is 150 seconds and population size for GA is 500. The stopping criteria for GA is self-maintained and trained to stop running when no changes in results is found for 1500 generations. For the multi-robot approach, a total of three robots are considered and the performance parameters (e.g., power rating, constraints, processor details etc.) for each robot are presented in Table 5.2. In comparison to the robot processor, the cloud VM is considered to be a minimum \(M\) times faster than the fastest robot processor \((S_c = M \times S_{R_p})\). Finally, for communication modelling, an infrastructure using IEEE 802.11 WLANs is presented with a maximum bit rate of 54 Mbit/s and 8 available stream bit rate \((6, 9, 12, 18, 24, 36, 48, 54\, \text{Mbit/s})\), similar to the simulation in chapter 4. Based on this, the whole workspace is designed with four APs where each AP has three users. As explained in 5.4.1, each of the locations correspond to bandwidth value for a selected robot based on the selection of AP. All this information is available to robots at the time of operation and are taken into consideration for a parameter (i.e., energy, time, distance) calculation and for the complete decision-making set. Based on all of this, the simulation is run, and the results are verified after comparison with the above-mentioned reference methods.

Table 5.3: Analysis of GA-based decision-making scheme

| Task No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| Allocation | 1 | 0 | 0 | 3 | 0 | 2 | 1 | 0 | 3 | 3 | 3 | 1 | 0 | 3 | 0 | 3 | 3 | 3 | 0 | 2 |
| Offloading | * | 3 | 3 | * | 1 | * | * | * | 3 | * | * | * | 3 | * | 3 | * | * | * | 3 | * |
| Robot     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Location  | 1 | 7 | 21 | 27 | 27 | 31 | 31 | 27 | 30 | 30 | 35 | 35 | 21 | 16 | 8 | 15 | 21 | 21 | 21 | 21 |
| Access     | * | 1 | 4 | * | 3 | * | * | 3 | * | * | * | * | 2 | * | 1 | * | * | * | 3 | * |
| Point      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
5.6.1 Analysis of decision-making scheme

The results from the GA-based scheme is presented in Table 5.3. As seen in the table, the four layers present the four sets of decisions. Layer 1 presents the task allocation decisions where ‘1’, ‘2’, ‘3’ indicates the robot assigned to the task, whereas ‘0’ indicates the task has been offloaded to the cloud. Layer 2 points to the selection of the robot for offloading the tasks to the cloud. Henceforth, this is a subset of the allocation decisions, where tasks have only been offloaded ($I_{t_i} = '0'$). Tasks that are completed on the robot cannot be offloaded and hence are defined in layer 2 as ‘×’. Layer 3 points to the location decision. For tasks completed on robot, the location points to the cell in the workspace where the corresponding robot has completed the task.

For instance, [3 × 27 ×] for task 4 means that the task is completed by robot $R_3$ at cell 27. In contrast, for tasks that are offloaded, location points to the cells the task was offloaded from by the corresponding robot. For example, results for task 30 are presented as: [0 2 17 4] which means that task 30 is offloaded to cloud VM by robot $R_2$ from location 17. Finally, layer 4 of the results indicates the selection of AP, which along with location, corresponds to available bandwidth utilized by the robot. Similar to layer 2, this is a subset of allocation decisions where tasks have been offloaded. The tasks that are completed on the robot do not require any AP association and are hence denoted by ‘×’ in the results.
(a) Best Fitness measure (i.e., Lowest Energy)

(b) Average Fitness measure

(c) Task Allocation Stem

(d) Task Allocation Histogram

Figure 5.10: Fitness performance and offloading decisions of GAMRC scheme

i) Analysis of Fitness Score and Offloading Performance: Fig. 5.10 shows the performance of our proposed GA scheme (GAMRC) where the key indicator is fitness. According to Fig. 5.10a, the average fitness graph shows a declining trend, which signifies that our GA scheme is working properly. Since the objective is to find the minimum energy, the falling graph suggests that results go from initial findings towards lower over time as we start gaining more minimal/precise values of fitness measure. Similar findings is seen for the best fitness graph (Fig. 5.10b). As mentioned before, in the context of the application, the term “best fitness measure” indicates the lowest value of robot energy. From the initial generation, GA results are evaluated based on their
fitness measures. Over the course of the complete algorithm run, each time a lower fitness measure is found, it replaces the previous value and becomes the new best fitness measure. When the best fitness measure doesn’t change for a pre-defined number of generations, then the result is assumed to be near-optimal. Hence, the best fitness measure finishes as a constant line in the graph.

Fig. 5.10c is a representation of each task and its corresponding allocation. According to the graph, it is found that in cases of parallel tasks, allocations are shared between robots and the cloud VM. It points to the proper sharing and utilization of resources for faster task completion. The same is seen from Fig. 5.10d where the allocation histogram indicates the well-distributed nature of the allocation results. Although approximately 14 tasks were completed by the primary robot $R_1$, robot $R_2$ and robot $R_3$ provided aid through local offloading along with the support from the cloud infrastructure. More specifically, it is evident from the results that robot $R_3$ was the biggest contributor in terms of the task allocation. But, this is a reasonable outcome since $R_3$ is the most powerful robot in the given scenario, and takes the majority of the additional workload of the application.

Table 5.4: Performance of each robot for GAMRC approach

<table>
<thead>
<tr>
<th></th>
<th>Robot $R_1$</th>
<th>Robot $R_2$</th>
<th>Robot $R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Constraint</td>
<td>5000 J</td>
<td>1000 J</td>
<td>3000 J</td>
</tr>
<tr>
<td>Energy Consumption</td>
<td>936.18 J</td>
<td>410.78 J</td>
<td>1489.53 J</td>
</tr>
<tr>
<td>Path Planning Results</td>
<td>1-27-31-35-5-24</td>
<td>1-31-21-17-4</td>
<td>1-7-21-27-30-35-24-16-8-15-21-10-4-11-4-16-17-24-17-15</td>
</tr>
<tr>
<td>Total Distance Covered</td>
<td>204.84 m</td>
<td>132.42 m</td>
<td>335.54 m</td>
</tr>
</tbody>
</table>
ii) Analysis of Individual Robot Performance: We present the performance of individual robots in Table 5.4. As seen, each robot has an energy limit, which also influences the offloading decisions. The near-optimal decision-set requires each robot to be within their energy constraint. As seen from the results, energy consumption of robot $R_1$ (936.18 J), $R_2$ (410.78 J) and $R_3$ (1489.53 J) are all within their respective energy constraints of 5000 J, 1000 J and 3000 J respectively. This outcome verifies that the algorithm identifies the near-optimal decision-set while meeting the energy constraint condition of each individual robot. It also highlights that $R_3$ carries most of the workload, which results in the highest energy consumption among them, whereas $R_2$ is the least utilized robot. All these results clearly indicate that robot $R_1$ offloaded tasks to the cloud VM in addition to the local robots ($R_2$ and $R_3$) for computation support as well as easier communication with the cloud.

iii) Analysis of Path Planning Performance: The path planning results from Table 5.3 shows the cells each robot has visited as well as the order in which it has visited the cells. The underlined cells in the results mean that offloading took place in these locations of the workspace. These results help prepare a path plan for an individual robot. Additionally, these outcomes also highlight the total distance covered by each robot.
According to the results, robot $R_3$ covers more area through movement and hence has a higher distance coverage (335.54 m) than $R_1$ (204.84 m) or $R_2$ (132.42 m). The underlined cells in the path planning results point to the cells that the robots have visited for the purpose of task offloading. It is further evident from Fig. 5.11, which shows the path plan of each robot for our proposed GAMRC scheme. In these graphs, the continuous directed lines help trace the path of each robot. The dotted lines mean that the robot communicated with the cloud for offloading during its stay in these cells. Finally, the non-directed continuous lines indicate an intermediate path while it was moving towards a selected cell.

The path planning results highlight a clear view of each robot’s movement for the duration of the application. These results also help relate to the access point (AP) selection and therefore the overall task offloading performance, since movement has an impact on the available bandwidth each robot gains at different locations for cloud-based communication.
iv) **Analysis of AP Usage:** Depending on the task allocation and path plan results, the robots are allotted to different locations for task completion as well as for offloading during the course of the operation. Based on these decisions, each robot has different availability of bandwidth, which also influences the AP selection decisions. Fig. 5.12 depicts the AP selection for each task. The results show that a total of four tasks are offloaded using AP 4 and a total of three tasks are offloaded using AP 3, while three tasks are also offloaded via selection of AP 2. Finally, selection of AP 1 results in a total of four tasks being offloaded. This attains to 14 uses of APs, meaning a total of 14 tasks have been offloaded to the cloud. We further gather from Table 5.3 that a total of 11 of these tasks have been offloaded by robot $R_3$, which makes it the dominant robot in the case of offloading to the cloud. In comparison, the performance of $R_1$ and $R_2$ are meagre as they offload only 1 and 2 tasks respectively. Accordingly, the findings from this section suggest that the ability of robot $R_3$ to cover more distance results in better access to available bandwidth (through AP selection), which results in robot $R_1$ getting aid from $R_3$ for the majority of task offloading during application. This ultimately increases the potential of improving the system outcome.
Figure 5.13: Path plan performance comparison of robot $R_1$ using all three methods
Table 5.5: Task offloading performance comparison (Min: $E_{\text{total}}$)

<table>
<thead>
<tr>
<th>Result Parameters</th>
<th>GA (Multi-Robot with Cloud), GAMRC</th>
<th>GA (Single Robot with Cloud), GASRC</th>
<th>GA (Multi-Robot on-Board), GAMRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation No</td>
<td>11524</td>
<td>8399</td>
<td>N/A</td>
</tr>
<tr>
<td>Offloaded Task (Cloud)</td>
<td>14</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Minimal Energy</td>
<td>2836.50 J</td>
<td>4778.80 J</td>
<td>7057.19 J</td>
</tr>
<tr>
<td>Total Time</td>
<td>80.31 sec</td>
<td>109.45 sec</td>
<td>413.16 sec</td>
</tr>
<tr>
<td>Total Distance</td>
<td>672.8 m</td>
<td>297.98 m</td>
<td>1161.96 m</td>
</tr>
</tbody>
</table>

5.6.2 Comparative performance evaluation

In Table 5.5, we evaluate the performance of our GA scheme for multi-robot and cloud (GAMRC) with results from the GA scheme for a single robot and cloud (GASRC) and the GA scheme for multi-robot on-board (GAMRB). The results clearly highlight that GAMRC consumes much lower energy (2836.50 J) than GASRC (4778.80 J) and GAMRB (7057.19 J). In terms of time/delay, GAMRB doesn’t finish within the time constraint. As for GASRC, even though the tasks are finished within the delay constraint, the time is still higher than the GAMRC process, which is the fastest to complete the applications. The reason for the better performance by GASRC and GAMRC is identified via a deeper analysis.

i) Offloading Performance Comparison: According to Table 5.5, GASRC and GAMRC entail lower energy because of their ability to offload more tasks to the cloud. Here the benefit of GASRC is evident, since the single robot $R_1$ allows for the most
offloading of tasks (21) to the cloud. Despite the fact that only 14 tasks are offloaded to the cloud for GAMRC, the added trait of local R-R communication means that robot $R_1$ utilizes nearby robots $R_2$ and $R_3$ for local completion of tasks or help with offloading of tasks to the cloud. This results in better access to resources (cloud VM and local robot) for GAMRC (w.r.t GASRC), more execution of parallel tasks and faster completion of tasks for cloud networked multi-robot systems. In comparison with results from chapter 3 and 4, the results also suggest that GAMRC performs well for increased nodes in DAG. So, it is a clear indication that GAMRC has the capability to deal with different complexities in task graphs, which is an important feature of GA. As for the GAMRC overhead, simulation results here has been considered only after it manages to meet the time deadline after the inclusion of overhead (similar to previous chapters). The results suggest that even with increased number of decision-layers (robots, APs, local and cloud-based offloading), the modified GA approach can meet the system criteria and perform in an optimal manner, even though the overall completion time is expected to be higher than chapter 3 and 4 due to increased number of decision-layers (4-layers).

**ii) Path Planning Comparison:** According to Table 5.5, the total distance covered by GAMRC (672.8 m) is higher than GASRC (297.98 m). Even though more movement causes higher energy consumption, for GAMRC there are three active robots that are moving in tandem to get access to better bandwidth for easier offloading. Besides, the robots are also utilizing their local communication to cover more area effectively. Hence, the primary robot $R_1$ gets assistance from robot $R_2$ and robot $R_3$ for completing local tasks or to help with communication for cloud-based offloading. Unfortunately for GAMRB, the lack of cloud availability hampers the system performance. Even though the robots cover the most distance (1161.96 m), it doesn’t provide much benefit as some tasks are too latent and consume high energy in the local processor. Here the additional movement
Table 5.6: Comparison of robots’ performance among the three methods (Min: $E_{total}$)

<table>
<thead>
<tr>
<th>Robots</th>
<th>GA (Multi-Robot with Cloud), GAMRC</th>
<th>GA (Single Robot with Cloud), GASRC</th>
<th>GA (Multi-Robot On-Board), GAMRB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_1$</td>
<td>$R_2$</td>
<td>$R_3$</td>
</tr>
<tr>
<td>Energy Constraint (Limit)</td>
<td>5000 J</td>
<td>1000 J</td>
<td>3000 J</td>
</tr>
<tr>
<td>Energy Consumed</td>
<td>936.18 J</td>
<td>410.78 J</td>
<td>1489.53 J</td>
</tr>
</tbody>
</table>

may utilize the robots more, but the robot’s local processors can’t compensate for the high computational requirements of these tasks. In comparison, for GAMRC, the robots offload such tasks to the cloud and even move accordingly to help with offloading. This saves valuable time and energy, as reflected in the performance. Finally, an analysis of movement for robot $R_1$ (Fig. 5.13) also reaches the same conclusion, where additional robots and cloud assistance results in robot $R_1$ moving less (204.84 m) in GAMRC than in the other two validated methods of GASRC (297.98 m) and GAMRB (582.7 m).

iii) **Performance Comparison of Robot**: Given that each robot has its own energy constraint, proper utilization of energy is a top priority in such applications. Due to the involvement of the cloud, the energy used for each robot was within the energy limitation (as seen in Table 5.6) for our GAMRC approach. In comparison, the GASRC process that runs the operation with robot $R_1$, entails an energy of 4778.80 J, which is significantly higher than for the single robot scenario, even though it is slightly under the energy bounds of 5000 J. The same can be said about the GAMRB process, which results in a higher value of total energy as well as higher energy consumption for each of the robots.
(2532.2 J, 1610.7 J and 2914.21 J for $R_1$, $R_2$ and $R_3$ respectively). Among all three methods, the lowest energy consumption for the principal robot $R_1$ is for the GAMRC method (936.18 J). It indicates that even though all the methods manage to meet the individual robot’s energy constraint, the performance of the GA-based scheme for cloud networked multi-robot system results in the lowest energy consumption for each of the robots (including the primary robot $R_1$).

5.7 Summary

Task offloading-based decision-making is a critical issue for cloud computing in networked robotic applications. Contrary to single robot applications, the offloading for a cloud networked multi-robot system is further complicated by the addition of local R-R communication along with the cloud-based R-C offloading. Therefore, a proper balance of workload between local and cloud-based offloading is required. Moreover, the offloading performance is greatly influenced by the ability of the robot to move on-demand and gain access to better gateways (communication links) for connecting to the cloud. Consequently, in this chapter we have merged all these aspects and proposed a novel 4-layer decision-making scheme to identify all the near-optimal solutions for task offloading (in multi-robot systems) by leveraging the complementary strength of network connectivity, path planning and local robot-robot interaction. Based on our proposed framework (chapter 3) for task offloading in a cloud networked multi-robot system, the robots communicate with each other and offload tasks by utilizing their motion and connectivity features. Our joint optimization problem for offloading in a 36-cell workspace and 40-node taskflow is derived from the motivational application of parcel sorting and distribution in an automated warehouse. Based on this scenario, the optimization problem for offloading is tackled by a GA-based scheme that proposes 4-
layers of decision-making (task allocation, robot selection for offloading, movement decision and AP selection). The complete simulation results are then acquired and verified through comparison with two validated benchmarks: one that considers GA-based offloading for a single robot and another implements a method for multi-robot systems without any inclusion of the cloud. The outcome implies that our approach performs superior in terms of energy usage as well as completion time/latency. We also conclude from further investigation that the addition of the cloud helps complete the computation-heavy tasks more quickly. Besides, the local R-R sharing utilizes the available resources to offload tasks in a more efficient way. Overall, all the factors combine to attain a system performance that is more enhanced in every way than the other implemented procedures considered as benchmarks (from the previous chapter). Also, this proves that even though the offloading mechanism for a multi-robot system is more complicated in terms of making decisions, at the same time it has the potential to provide better results (if optimal decisions can be attained), which is why a separate/specific study on task offloading in this domain (multi-robot system) has been attempted. In the following chapter, we will summarize our contributions throughout the thesis as well as discuss some possible scope of future work.
Chapter 6.

Conclusion and Future Work

In this chapter, we recapitulate the main contributions made in this thesis. Furthermore, we discuss the possible open problems in this field as well as highlight the possible future research directions based on the thesis.

6.1 Summary of Contributions

Communication and mobility play key roles in task offloading decisions for CNR. While communication dictates the bandwidth for cloud communication, mobility decides the location where offloading is done as well as the path which the robot follows. Different from mobile cloud computing, the “on-demand mobility” for a robotic network implies that robot actively plans its path and selects the most suitable communication links (AP selection) to accommodate task offloading. Therefore, in this thesis, we have studied the unique relationship among these three parameters (task offloading, path planning and AP selection) in the context of CNR applications. To reiterate, the key contributions of the thesis are summarized as follows.

We considered two application scenarios: smart city and smart manufacturing. Initially in chapter 3, we presented an integrated framework of cloud, robot and wireless sensors that enables task offloading to the cloud. Based on this, we proposed a smart city crowd control application as a single DAG taskflow and formulated our offloading decision-making problem. During this formulation, we considered mobility and communications aspects as fixed values and designed a genetic algorithm-based (GA) scheme to identify
the optimal offloading decisions. Simulation results for two scenarios suggested the same thing, which is: the GA-based scheme identifies the near-optimal solution, but in less time and with more efficiency (compared to the exhaustive method, All on Robot approach and greedy approach). At the same time, this helped validate our results with respect to the already authenticated benchmarks. In addition, we also ran simulations by varying mobility parameters (distance between zones) as well as communication parameters (bandwidth), which suggested that both aspects have influence on the task offloading. In order to study its impact further, we ran a multi-taskflow path planning problem, where the robot had to choose the correct order of taskflow to complete the application. As part of this decision set, the robot’s movement decisions and available bandwidth (at chosen locations) are integrated with its offloading decisions. Simulation results suggest that task offloading is highly influenced by both mobility and communication. At the same time, the latter simulation advises that further decision-making approaches can be designed by integrating offloading with movement decisions and communication link selection.

This motivates our contribution in chapter 4 where we utilized the interdependent relationship among path planning, AP selection and task offloading to formulate a joint-optimization problem. For simulation, we consider a smart factory maintenance application where a 30-node DAG needs to be completed and the cloud-assisted robot needs to determine the near-optimal solution for all three factors (in order to minimize energy). A modified GA-based scheme was designed to solve the given problem set. The solution in this case included three layers of decisions for offloading (which task to offload, which location to complete a task, and which AP to select for offloading). As part of validation, a simulation was run to compare the results with a GA scheme with fixed movement and communication (from chapter 3). The outcome helped prove that variable movement and AP selection decisions potentially assist the robot to improve the
offloading decision further as well as the system performance. As the GA scheme designed in chapter 3 had already been validated with respect to the exhaustive and greedy methods, our results in this chapter present an improvement on top of the current status quo for a cloud-assisted robotic system.

In chapter 5, we considered a cloud networked multi-robot application where a group of robots worked together as part of the application. Contrary to a single robot application in chapter 4, task offloading decision process in this scenario is more complicated, hence this enquiry deserves its own study. In addition to the robot-cloud communication, the robot-robot communication needs to be taken into consideration as well. Therefore, we considered four parameters (offloading, path planning, AP selection and robot selection) for making decisions which helps formulate our joint optimization problem for a smart warehouse parcel management operation. In order to solve the problem, we further modified the GA properties and designed a scheme with four layers of decisions (which task to offload, which robot to offload task, which AP to select for cloud communication and which location to complete a task). As seen from the simulation results, the communication with nearby available robots led to superior system performance (compared to a single robot application in chapter 4), as tasks were offloaded with more ease and workload was more balanced among the additional entities. As a result, the system consumed significantly less energy and completed tasks faster in a cloud networked multi-robot system.

In summary, ours is the first approach to study the interdependent relation among offloading, path planning and AP selection in the context of CNR to design optimal task-offloading algorithms. This approach will pave the way for more integral decision-making approaches (related to offloading) by utilizing different aspects of cloud networked robotic applications. As for practical implementations, the findings of our
work led to the development of a SwinCBot [221] (a cloud-aided robotic hardware system) as a proof of concept that has provided some initial results in applications such as face recognition and path planning. Thus, it is in progress to bridge the gap between the theoretical findings and practical applications of offloading based decision-making in cloud-networked robotics.

<table>
<thead>
<tr>
<th>Model</th>
<th>Robustness</th>
<th>Interoperability</th>
<th>Mobility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer-based</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Proxy-based</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Clone-based</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Figure 6.1: Traits of computing models for cloud robotics (Hu et al., [11])

6.2 Recommendations for Future Work

Besides the contributions made in this thesis, the design of mobility-driven and communication-aware offloading in CNR applications sparks several ideas that forms the basis of several future research studies. Some of these are presented as follows:

6.2.1 Real-time resource allocation to deal with dynamic network structure

CNR is still a new area of research due to the on-demand mobility of robots. This presents several opportunities for innovation. One particular aspect is to design real-time resource allocation algorithms to deal with the dynamic network structure. For real-time applications, it is often difficult to map the cloud computing framework to multi-agent networks due to resource constraints such as limited bandwidth and dynamic structure that incurs high latency. In such applications, the robots need to guarantee a constant stream of data transmission (e.g., video, audio, map etc.) to the cloud data centre. As explained by Hu et al, [11] in Fig. 6.1, different cloud robotic models have different traits.
Hence, it is imperative to have a real-time resource allocation strategy where the objective for the robot is to choose from different computing models of the cloud robotic architecture in order to meet the application requirements. Robustness, interoperability and mobility are all key attributes when considered as performance parameters for peer-based, proxy-based and clone-based models. Because of that, robots may need to form different dynamic structures in order to maintain a constant stream of data transmission by modelling systems with optimal resource allocation. The existing algorithms (e.g., Stackelberg method [109], dynamic GA [151], greedy, auction-based method [93] etc.) serve as a good starting point in order to design novel algorithms for decreasing the undetermined complexity of these NP-hard problems so as to realize real-time communication. This is still an untouched area of study that has potential for researchers.

Figure 6.2: Architecture of cloud-fog computing system (Pham & Huh [222])
6.2.2 Task allocation for cloud robotic services by introducing edge resources

To further utilize the mobility and communication aspects of multi-agent services in real-time applications and ensure proper data transmission between robots and cloud, the concept of edge resources has been introduced in the literature. In our case, the introduction of edge resources helps robots communicate with the cloud much faster through a middleware. There are different types of edge components that are worth exploring for robotic applications. We present two cases in this space:

![Diagram of cloudlet-based mobile cloud computing system](image)

Figure 6.3: A model of cloudlet-based mobile cloud computing system (collected from Liu et al. [223])

a) Fog computing (Fig. 6.2), also known as “clouds at the edge,” is a developing paradigm that allocates services near the devices to improve the quality of service (QoS). Task allocation studies using fog computing are already available for mobile cloud computing. For CNR though, the “on-demand mobility” (explained in this thesis) of robots make the scenario more dynamic, which is why a thorough research is required.
b) Cloudlet [224] infrastructure is another option that may be introduced in between robot–cloud communication. Cloudlet is widely adopted as an extension of the cloud closer to the data source which provides multi-robot systems with virtualized resources to execute the latency sensitive services. Accordingly, a similar type of resource allocation problem can also be studied for CNR, where robot’s path planning and offloading will also consider cloudlet accessibility in its decision-making for various cases (Fig. 6.3).

As Pham and Huh [222] and Liu et al. [223] have already proposed allocation for mobile cloud computing by introducing edge resources and cloudlet infrastructure. However, the scenario is more dynamic for CNR from the application point-of-view, as robots have on-demand mobility. In such cases, there is scope for algorithm design, where these new resources (edge/cloudlet) are part of a multi-objective optimization problem along with offloading and path planning in the context of CNR.

![VM migration patterns in a MCC system](image)

Figure 6.4: VM migration patterns in a MCC system (Gkatzikis & Koutsopoulos [225])

6.2.3 Virtual machine (VM) migration in cloud networked robotics

While cloud computing in networked robotics minimizes the response time of an offloaded application, it is important to focus on the cloud aspects of task offloading
decision-making, where the key aspect is the performance of cloud virtual machines (VM) and their deployment. A challenging aspect of VM deployment is the additional computing resource usage management of VM which requires computing resources for VM creation and configuration [226]. More specifically, the performance is managed by VM migration which brings the cloud resources closer to the user [227]. As part of VM migration, current-hosted workload is migrated from one server to another, which improves the system outcome but trades-off in terms of complexity.

Contrary to the previous studies that mostly put emphasis on multi-robot coordination and the cloud-robot interface, the cloud aspects of the study have the potential to concentrate on developing novel cloud architectures and migration mechanisms. By doing so, this approach will overcome the dynamic nature of the cloud-based robotic applications where locations change due to on-demand mobility and bandwidth of the access links change with time. Therefore, there is ample scope for researchers to exploit the nature of cloud-aided robots to prepare a centralized framework of task offloading that includes allocation, movement and bandwidth as well as VM scheduling and migration patterns as part of its decision-making.
References


<table>
<thead>
<tr>
<th>No.</th>
<th>Reference</th>
</tr>
</thead>
</table>


Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, April 2015, pp. 1894-1902.


Interventions and Environmental Surveillance (RISE), Benicassim, Spain, January 2008, pp. 1-17.


