# Optimal Placement of Controllers in Software Defined Networks

by

**Afrim Sallahi**, **BIT**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

**Master of Science in Information and Systems Science**

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
May, 2014

# Optimal Placement of Controllers in Software Defined Networks

submitted by **Afrim Sallahi**, **BIT**

in partial fulfillment of the requirements for the degree of

**Master of Science in Information and Systems Science**

Professor Marc St-Hilaire, Thesis Supervisor

Professor Roshdy Hafez, Chair,
Department of Systems and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
May, 2014

# Abstract

This thesis focuses on deriving exact methods to place controllers on a network for the planning problem of Software Defined Network (SDN). A first mathematical model, referred to as the planning model, is proposed that decides the optimal location to place controllers on a network while minimizing the overall cost. A second mathematical model, referred to as the expansion model, is proposed that determines the changes to optimally place controllers on an existing SDN network. The advantages of the proposed models are to incorporate realistic constrains. A solver optimizes the exact method for both models with many scenarios and the results are compared. In the planning problem, as the input size increased, the time to find the optimal solution also increased in non-polynomial time. The expansion problem results show that changes can be made to existing implementations by altering many of the input variables. The expansion model allows control of how much of the existing network changes.

To my mother and brothers.
May they rest in peace.

# Acknowledgments

I would like to express my very great appreciation to Professor Marc St-Hilaire for his constructive suggestions during the planning and development of this thesis. His patience, devotion and guidance has helped me get here today. It has been my great honour and privilege to work under your supervision.

I would like to thank Dr. Georgina Fitzgerald for proofreading my thesis. I would also like to extend my thanks Dr. Blerim Qela for his guidence.

I am particularly grateful for the assistance given by my brother Arton. Finally, I would like to thank the rest of my family for their support throughout my study.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| **API** | Application Programming Interface. 10, 11, 13, 14 |
| **ForCES** | Forwarding and Control Element Separation. 11 |
| **IP** | Internet Protocol. 1, 26 |
| **LTE** | Long Term Evolution. 14 |
| **OS** | Operating Sytem. 9, 11–14 |
| **OSPF** | Open-Shortest Path First. 10, 13 |
| **QoS** | Quality of Service. 4, 15 |
| **RSVP** | Resource Reservation Protocol. 9 |
| **SDN** | Software Defined Network. iii, 1–7, 9, 11–20, 28, 62, 65, 66, 72, 82 |
| **SSL** | Secure Socket Layer. 11 |
| **TCAM** | Ternary Content-Addressable Memory. 8, 11, 12, 15 |
| **TCP** | Transport Control Protocol. 1, 26 |
| **VLAN** | Virtual Local Area Network. 15 |
| **Wi-Fi** | Wireless Fidelity. 13 |

# Chapter 1

# Introduction

Today, computer networks are used extensively to enable communication throughout the globe. Only recently, in North America, the dominance of real time entertainment (comprised of streaming video and audio) continues to be the greatest impact on computer networks and it will only continue to grow in the future. Other areas that had an effect on networks are cloud based computing and storage. With such changes, fixed and cellular networks have to be able to be very adaptable on their network infrastructure. A resolution to this is moving current networks to Software Defined Network (SDN).

Two universities using packet switched networking method made the first end-to-end communication between two hosts in 1975. The Transport Control Protocol (TCP) and Internet Protocol (IP) were the two protocols that made it possible to communicate between these two hosts. Two years later, a computer in United States, United Kingdom and Norway were able to communicate between each other using the same protocol (TCP/IP). Hardware and software had evolved and networking equipment had grown rapidly in terms of number of protocols, performance and pricing to support the demand. Unfortunately, innovation is very hard to do because the network equipment is closed and proprietary. For example, prototyping new protocols can not be done due to the fact that the ecosystem of network elements is closed.

In the networking field, SDN tries to abstract the networking equipment and its operating system. The two functions networks perform are control and switching where the former makes decisions where information goes and the latter is responsible for moving information hop-by-hop. The switching plane is kept the same but the control plane is abstracted and moved into a central location within the network and it is called the “controller”. A network is easiest to adapt when it is designed properly and in this thesis, we are interested in the optimal placement of controller(s) in SDN networks.

## 1.1 Problem Statement

The single controller within a network has many benefits, one of them is that the controller has a single view of the whole network and algorithms such as routing are performed using that view but disadvantages still exist. The problem with a single controller is that it is the only controller in the network - network traffic for the most part is known to be bursty and a controller may get overloaded. Even a single controller sometimes is hard to place in a network because one needs to know where to place it. That particularly holds when the network includes multiple geographical regions as one region might have more traffic than the other during different times of day. For example, employees make use of the network during the day in a given region and backups run at night in another region. A single controller has problems with redundancy, load balancing and local events become global to the controller. A solution to the single controller problem is to use multiple controllers. There is a gap in research related to multiple controllers because most of the studies have done work with only one controller.

Having multiple controllers solves problems with redundancy, load balancing or separating areas in its own domains so the events can be handled locally. Despite its benefits, using multiple controllers also comes with problems. One big problem is, again, the placement of such controllers. The question then becomes: how to place such controllers on a network (and possibly how many are needed). The answer to this question relies on the network characteristics themselves and what is chosen to “measure” when deciding where to place the controllers. Furthermore, the placement method of controllers can be made to be exact or approximate - both of which have advantages and disadvantages.

Even after the planning of controllers is complete and networks are implemented, over time, the networks experience growth and require changes. When a network already has controllers installed, currently, there are no methods for current networks to expand while considering already installed controllers. This means that the method that was used to plan the controllers initially cannot be used again to expand a network. This is due to the fact that changing a network infrastructure is very expensive and a planning model would completely change a network. Therefore, the expansion of a network is problematic in terms of placement of controllers and a proper solution needs to be found.

Placement of controllers has already been researched but the research involves measurement techniques that only rely on latency. Having more measurements within the placement calculation makes it possible to include more concerns when placing controllers optimally on a network. Besides not minimizing the cost of controller placements, some of the concerns that have not been addressed in current research are: link bandwidth between switches and controllers, controller and link inventory, flow-setup latency and controller to controller connectivity. This is something that should be addressed because it would ensure better planning results for new networks. In this thesis, the aforementioned concerns will be included with proposed exact mathematical models for optimal placement of controllers on a new network (i.e.green field scenario) or an existing SDN network.

## 1.2 Research Objectives and Contributions

Based on the problem statement above, the main objective of this thesis is to model problems that find the optimal placement of controllers on a network. The ultimate goal is to contribute to the field of SDN by shedding more light into the controller placement where different mathematical models are proposed with more realistic constrains. As a result, to advance the acceptance of SDN methodology into current networks, the following contributions are made:

- Propose a mathematical model to plan SDN networks starting from a green field scenario. Given a set of switches that must be managed by the controller(s), the model simultaneously determines the optimal number, location, and type of controller(s) as well as the interconnections between all the network elements. The goal of the model is to minimize the cost of the SDN while considering different constraints such as the capacity of the controller(s), capacity of links between controllers and switches, the flow-setup latency.

- Evaluate the performance of the planning model to make sure it behaves correctly and analyze the results given different scenarios are optimized. The following scenarios are considered:

  - Generate many topologies that start from a small topology size and increase to a large topology size.
  - Generate other topologies that determine how the number of possible controller locations affect the optimality.

- The goal is to generalize the planning model proposed in the planning model so that existing infrastructure (if any) can be considered in the planning. This generalized model can be used to plan a new network and update existing ones. The model is able to have different effects on the existing SDN network by defining a cost for making changes on an existing network.

- Evaluate the expansion of a network to make sure it behaves correctly and analyze the results given different change scenarios. The following scenarios are considered:

  - Expand multiple existing SDN networks by adding or removing switches to evaluate the performance and the cost of the expansion.
  - Expand an existing SDN network by adding switches and having the cost to remove existing network items that range from high to low.

The research objectives can be completed using the methodology mentioned in the next section.

## 1.3 Methodology

Before methodology is listed, the current network architecture is reviewed in order to better understand the new concepts. Few disadvantages are listed for the current network architecture. The SDN architecture is reviewed in order to compare to what has changed from an operating system's point of view. A literature on the application of SDN is performed in the areas of tooling, virtualization, routing and QoS, controllers and controller placement. For the controller placement, we review current research and list its limitations.

The planning of controllers in an SDN network was approached using the following methodology.

**Define the controller planning problem:**

In order to define the controller problem, we wrote a set of mathematical equations in order to represent the behaviour of SDN networks. Different constraints like controller capacity, controller and link types, bandwidth between switch and controllers and connectivity of the items in the network are considered since these are important when designing a SDN networks. The goal of the model is to minimize the cost while determining the number and the types of controllers to place in a network. To solve the model, we used a solver called CPLEX. The solver returns the optimal solution or the best solution found if a time limit is reached. The solution contains the location and type of controllers that were installed.

**Study the results of the planning problem:**

A detailed example explains the planning model. Then we optimize using CPLEX different topologies to analyze how long the solution takes, and how much the solution costs. The first result analysis starts with a small sample size and we grow it to large sample size to investigate how the model reacts. The second result analysis investigates how the cost is affected by having more controller placement locations for the same number of switches.

**Define the expansion controller planning model:**

Since we are inclined to include an expansion model to keep the existing network the same as much as possible, we had to add another costs to the objective function. The objective function is composed of the same costs as the planning model but with extra costs that include removing existing items. Conditional constraints are added in addition to what the planning model contains that allow the model to make better informed decisions when finding optimal solutions. We optimize the model using the same CPLEX solver that is used in the planning model. The solver returns the optimal cost and time taken to find the solutions. Furthermore, we are able to track how many switches have been removed or added (given that an existing SDN was optimized). The number of links that were removed and then planned again are also included in the model solution.

**Study the results of the expansion model:**

The expansion result and analysis starts with a detailed example that shows how the expansion model is applied. We investigate how the model reacts to expansions by adding or removing switches. Then, we explore how much of an effect the cost to remove items from the existing network has on the optimal solutions of the expansion model. The results are optimized using CPLEX

## 1.4 Thesis Overview

In this section, an overview of the remainder of the thesis is provided. In Chapter 2, the literature review shows the wide use of SDN. It also include a comparison between the current network architecture and the architecture proposed by SDN. Chapter 3 starts with the introduction of the planning problem followed by the mathematical model for planning controllers on a network. Then, the expansion problem is introduced followed by the mathematical model for expanding an existing network. The two models are optimized with a different set of topologies and the result is presented in Chapter 4. The chapter starts with the results for the planning model followed by the expansion problem. Finally, Chapter 5 summarizes the findings and limitations and proposes future research directions.

Chapter 2

# Background and Related Work

In this chapter, we review current and future network architectures and give their advantages and disadvantages. In Section 2.1, we review how the control and data planes currently work on the existing networking architectures. In Section 2.2, we introduce the SDN architecture. In Section 2.3, we review research focusing on planning and design of SDNs and finally, Section 2.4 concludes the chapter.

## 2.1 Current Network Architecture

It is important to review the current network architecture to better understand the new changes proposed. Current network architecture consists of transmission equipment, connectivity between components (wired or wireless), software and communication protocols. For network connectivity, different mediums connect hardware and of those connections, different transmission modes exist. Such transmission modes are: Ethernet, Wireless or Optical. The core of the network is then made of hardware (switches or routers) that is connected together with one or more transmission methods.

When we consider an end-to-end communication, the information travels through the network equipment hop-by-hop until it arrives at its destination. Sometimes, the information is segmented into multiple pieces and is sent through different paths to get to its destination. Routers and switches are devices that are stationed in the network to help with the information that is being sent around the network. They are very sophisticated devices because they decide what, where and how the information should go through the network. From here on, a router or switch that makes decisions on a network will be called a network element.

Deciding where the information should go is not straightforward because the devices require having knowledge about whom else is in the network. This type of information is managed by protocols that are installed in a network element. The control plane is a network element that decides where the information is coming from, what type of information it is and how to reach the destination. These protocols communicate with their neighbours and advertise information to keep a concise view of the whole network. Such protocols install rules on the network elements on Ternary Content-Addressable Memory (TCAM) tables.

Since we already know where the information goes, the next step is to know how and where to send such information. In each of the network elements, the TCAM table is looked up for the next hop information for that particular instance. Once the next hop port is found, it is just a matter of sending such information on that port for transmission. This is called the forwarding plane because of the physical operation it takes to look for information and move it to the next network element.

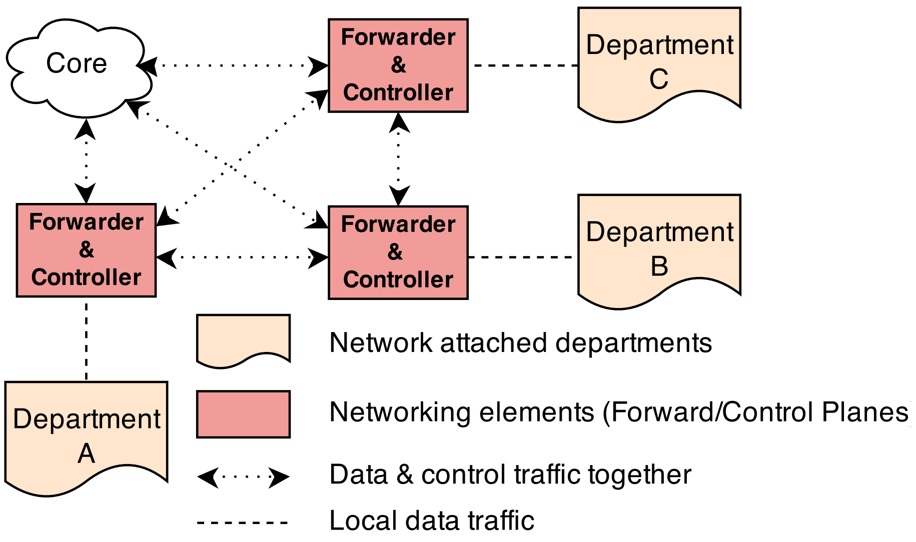


**Figure 2.1:** The current network that is deployed today. Data and control information is mixed within the core.

Figure 2.1 shows the current architecture of a simple network. In this example, we have the different departments accessing the network through network elements. Each of the network elements performs the forwarding plane and the control plane operations. Within the core of the network, the data path also contains control information from the network elements from its protocols. Next, some of the disadvantages that will be discussed are: limited innovation, difficult management, and costly equipment.

#### Limited innovation

One of the biggest issues that the current architecture faces is that it is a closed system. Networking equipment manufacturers have their own hardware, operating system, implementation of standards and their own extended set of features. This means that new protocols cannot be tested and verified in a timely fashion. This is due to the fact that only the manufacturer has access to the system. In the research community, one who wishes to test a new protocol cannot do so. The time taken when requesting new protocols with the networking equipment companies is very long because they are the ones who develop, verify and finally release it with the proprietary OS.

#### Difficult management

The operating system in such devices is closed for outside use. Besides Resource Reservation Protocol (RSVP) and its own management console, there is no other method to command the networking equipment. The operating system does not expose access to outside to command its internal system and this is a disadvantage because only two methods to manage it exist (RSVP and console). When a large network has proprietary hardware from different vendors, it makes it difficult to manage.

#### Costly

The equipment is extremely expensive. This is due to the fact that the cost of each network element includes time spent on developing and verifying the protocols, which is a lengthy process. For example, it is impossible to get networking equipment with only one protocol implemented by a particular manufacturer. The whole OS will be included for that particular model of hardware and its supported features. This drives costs high.

## 2.2 Introduction to SDN

The basic concept of SDN is to separate the control plane from the forwarding plane and provide abstraction between the controller and the network elements. Thus indicating that when a controller performs commands on a network element, the controller does not have to know the network element layers. Since every network element performs the same control plane procedures, it would make sense to move the "brain behind" the network (control plane) to a single location. The control plane is now on a centralized location and it is called "Controller". The controller then communicates with the networking elements by exposed Application Programming Interface (API). The controller is responsible for installing rules to a network element through that API.

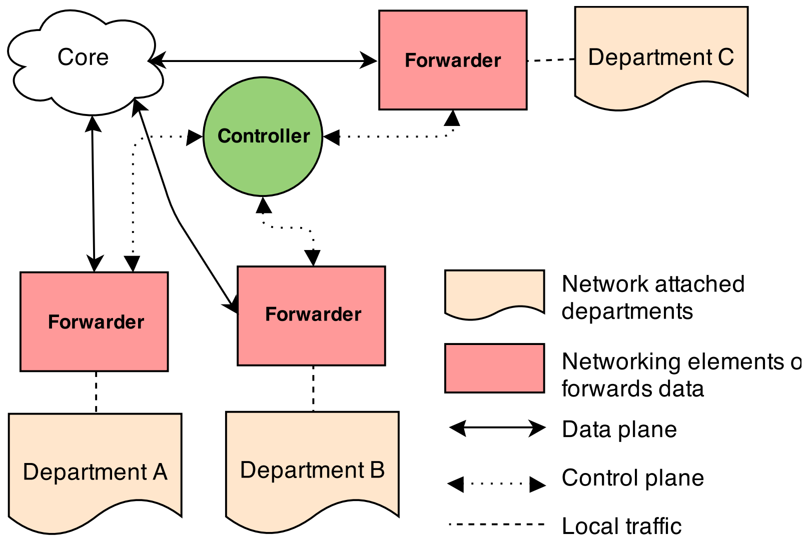


**Figure 2.2:** Typical SDN network architecture. Data and control information is separate. The controller keeps a global view of the network.

Figure 2.2 shows the new changes to the same network layout shown in Figure 2.1. The difference here is that a controller is placed and the networking elements communicate with it through different transmission methods. The controller probes the network elements to find the topology of the network. Based on this, the controller has enough information for a protocol to run. Such different protocols that are within the controller are called applications. Applications generate flows and the controller installs the flows to each of the network elements. For example, an application that is well known is the Open-Shortest Path First (OSPF) protocol. When OSPF is installed into a controller, it generates routing rules, and then installs them throughout the network.

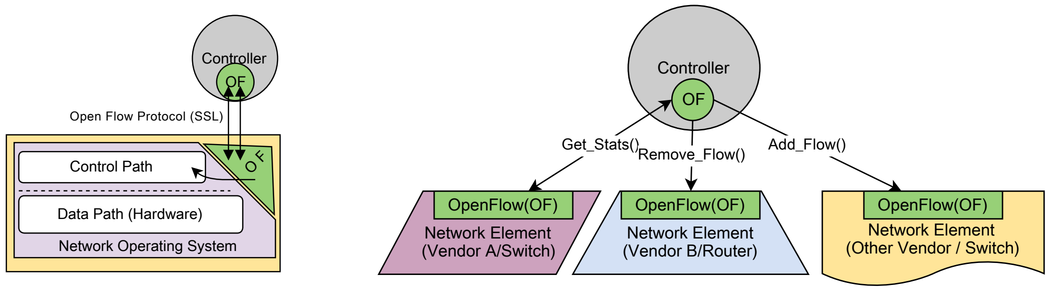
With SDN, the disadvantages mentioned above no longer exist. SDN invites innovations due to its exposed interfaces of the controller. Similarly, management will not be an issue since the rules that controllers install remotely are automatically translated to the network elements' rules by its own OS. Finally, the costs should be lower for networking elements because protocols are no longer needed and a network element comes as bare-bone as possible with just the OS. The applications running on the controller will be the replacement of the protocols in the current network elements. These protocols can be purchased as needed or be found free from the open source community.

### 2.2.1 Control Plane Communications

The communication between controller and network elements will be reviewed in this section. The question of how is it possible for multiple vendor equipment to coexist with the same controlling unit (the controller) will also be discussed in this section.

The two most promising standards that exist today that define controller protocol communication to network element are OpenFlow [1] and Forwarding and Control Element Separation (ForCES) [2]. Since OpenFlow is already implemented and is simpler to understand, we will be referring to this standard when explaining SDN concepts.

Abstraction is something that SDN aims at bringing to current networks and this means that common communication modules exist in each controller and network element. The network element implements OpenFlow standard and so does the controller. The calls that the controller makes to each of the network elements are done using API that are understood by the OS of the network element. For the network element's OS, these commands are translated into their own system specific rules and are most likely installed into TCAM table(s). Figure 2.3 shows such abstraction. Part A of Figure 2.3 shows the instance of OpenFlow within the network element's OS. The network element handles the controller communication using the module that implements the OpenFlow. In this case, both the controller and network element have the same OpenFlow standard. Once the network element implementation of OpenFlow completes communication with the controller, it then translates flow tables to its own rules and installs them to the local system. OpenFlow's specification indicates that the protocol must establish connection and run over Secure Socket Layer (SSL) for the controller to communicate with networking elements.



(a) System View (b) Network View

**Figure 2.3:** SDN views: (a) system view showing OS and hardware level and (b) network view with multiple network elements.

When we consider a network view as shown in part B of Figure 2.3, we can see, in a higher level, how the abstraction applies to the controller and network elements. The figure shows three different manufacturers translating and performing actions that the controller has requested on each network element. Network element A needs to perform and respond to the request `Get_Stats()`, this is a call that would get statistics from the network element and return them to the controller. Similarly, the second network element (from vendor B) is requested to remove a flow. In this instance, the OS would translate to an action where one or more entries from the TCAM table would be removed. Finally, the last example with another vendor shows that the controller performs the action to add an entry to the TCAM table.

The translation of rules between the OpenFlow protocol and the network element is the responsibility of the vendor amd must be implemented in the OS. Each switch reports its capabilities and the standard version it has implemented of OpenFlow. Based on OpenFlow 1.1 specification, Table 2.1 shows some of the possible fields that can be matched by a network element when processing the flows. With the fields shown in the table, any network element can be made to have capabilities of a switch, router, firewall, deep packet inspector, balancer, and many other functions that currently exist in multiple equipment types.

Another important aspect to uncover here is the acceleration of innovation. If we

**Table 2.1:** A flow table that shows different scenarios of flows that can be installed into network element.

| | Action | Src MAC | Dest MAC | Eth Type | VLAN ID | Src IP | Dst IP | IP Proto | IP Src Port | IP Dst Port |
|---|---|---|---|---|---|---|---|---|---|---|
| Flow Switch | Port3 | AA:EE | AA:BB | 0800 | 12 | 1.2.3.4 | 4.5.6.7 | 6 | 80 | 80 |
| Ethernet | Local | 15:AE | * | * | * | * | * | * | * | * |
| Routing | Port5 | * | * | * | * | * | 9.9.9.9 | * | * | * |
| NAT | Port2 | * | * | * | * | 1.2.3.4 | * | * | * | * |
| Firewall | Drop | * | * | * | * | * | * | * | 1214 | 1214 |
| Load Balancer | Port8 Port9 | * | * | * | * | * | 2.2.2.3 | * | * | * |
| Controller | * | * | * | * | * | * | * | * | * | * |

wish to have the OSPF routing algorithm on SDN, then we would have to implement the protocol in software in the controller. Since the controller has a global view of the network, running OSPF is simply done in one place. By running OSPF on the controller, the protocol would build the shortest paths and generate the flow tables for each route. Each of the flows are then installed on the correct network element by the controller using the available API. Here, when implementing the OSPF protocol into the controller, there is no need to know how to communicate with the networking elements and understand the operating system. This should accelerate innovation in the networking field.

## 2.3 Current Research on SDN

There are many organizations, companies, and universities that are focused on deriving standards, frameworks and tooling for SDN. Open Network Foundation is responsible for analyzing SDN requirements, and evolving the OpenFlow Standard. There is the OpenDaylight project that focuses on community-led and industry-supported open source framework for the SDN paradigm. There are companies such as Cisco, NEC, Brocade and others who have shown support for SDN by implementing various controller protocols that are available.

### 2.3.1 SDN in different envoirnments

SDN has been picked up by many of the different areas of the networking sector, especially the communication section. This section shows efforts made in introducing SDN into different fields of communication networks. This ranges from wireless in personal Wireless Fidelity (Wi-Fi) and sensor networks, to wired with ethernet, optical and finally to carrier networks. Ever since the introduction of pure packet networking in the cellular networks of Long Term Evolution (LTE), efforts are being made to improve the networking aspect of it [3–5]. The authors in [6] look into ways that SDN will help with the sensor networks when dealing with sensor nodes because the nodes are vulnerable and special care has to be taken. Work has been done to include optical networks in [7–9]. Effort in combining packet and circuit switched networking is done in [10, 11]. The authors in [12] looked at integrating optical into the SDN controller standard of OpenFlow. In [13], the authors manage to experimentally evaluate and converge packet and circuit network setup. This is a huge improvement because many physical layers can coexist within the same network and can be managed by the same controller. As we can see, although SDN is a relatively new technology, it is being used in many different environments and for various applications.

### 2.3.2 Tooling

Opening the current OS platforms through APIs brings many benefits to the networking area. Since programmability is involved in advancing the SDN landscape by being able to program protocols into the controller, then the right tooling is required to help with the process. As a solution is developed, debugging is used extensively to find problems that sometimes cannot be found. The authors in [14] have developed a debug tool that can log packet backtraces and set breakpoints in an SDN network. Verification is also an essential tool for providing error free solutions and the authors in [15] build a tool called No bugs In Controller Execution (NICE). They generate streams of packets of different events so that the controller undergoes real-time testing before it is put on a network. The authors in [16] use an algebraic symbolic verification algorithm where they model the flow tables operation and behaviours. The authors in [17] have developed Mininet. Mininet is a tool that is used to virtualize hosts, switches and topologies on computing power as small as a laptop. Here, the ability to prototype, implement and test a solution on many full-scale network topologies is possible. Once a solution is verified with Mininet, minimal to no changes are required to deploy the solution on a real SDN network. When considering testing the networks and security is involved, the authors in [18] have written Future Internet Testbed with Security (FITS) as a testbed for SDN networks. The authors in [19] have prototyped a tool that helps detect hidden inconsistencies in the network by exploring integrated network behaviour of real switch execution.

### 2.3.3 Virtualization

Besides the actual tooling, work has been done on the virtualization area of networking. Virtualization has matured throughout the years with intensive research into the areas of cloud computing. Interesting problems in the virtualization fields have arisen and SDN is already seen as a method of solving such problems. For instance, a bandwidth intensive task in a set of hundreds of virtualized hardware is to copy or move the virtual machines around the network. A solution is to find the best paths to move such traffic around the network with the help of SDN. The problem with backing up or moving virtual machines is investigated in [20, 21]. In [22], the authors introduce an SDN platform designed for network virtualization and their experimental results show that the tool can deliver different types of virtualization services.

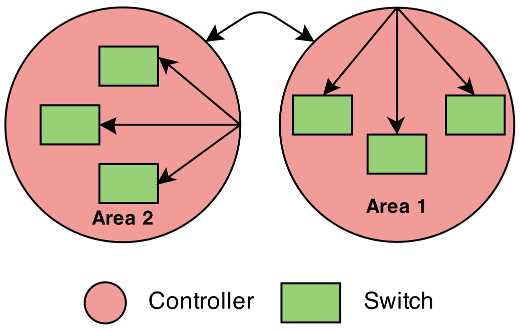
### 2.3.4 Routing and QoS

In relevance to virtualization, routing helped find better paths in copying virtual machines. Then, routers were thought of as just software and ideas of virtualizing router functionality came along in [23]. The virtual router would reside in memory and the rules would be installed in the TCAM tables where the lookup performance would not be affected. Furthermore, router as a service was proposed with its architecture and algorithms [24]. In [25], the authors propose a routing technique that adds the path to the source packet(s) and using this routing path, the packet traverses the network. Because the path is already computed and attached to the packets, there will be less communication with the controller, thereby reducing the controller traffic by 50% [25]. Different from routing, which focuses on available paths, quality of service is to provide guaranteed availability of resources. In [26], the virtualization of switches performs bandwidth isolation by marking Virtual Local Area Network (VLAN) priority bits in packets. The problem here is that the VLAN priority field has a length of three bits, which results in eight possibilities. The authors in [27] propose a feature to the controller that provides quality of service based on per-flow rate limiters and priority assignment. Some of the interesting solutions to QoS in SDN relate to scheduling algorithms and max-min fair share of traffic shaping [28]. Minimum cost flow optimizations are applied to adaptive video streaming in [29, 30].

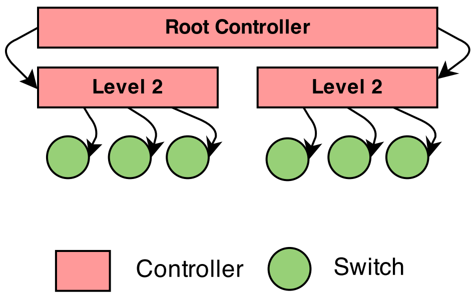
### 2.3.5 Controllers

The controller will face many technical challenges and since it is the brain of the new network paradigm, it should be looked at in more detail. Today, there exists many open source controllers that are available for many platforms and in many programming languages. They are: NOX [31], POX [31], Trema [32], SNAC, Flowvisor [26], Beacon and Floodlight [33]. Similarly, commercial controllers are now available. There has been research done in the controller itself and such work areas include: performance and capability, redundancy and distributed communication between controllers in the scenario of multiple controllers. In [34], the authors test the performance of the controller on a single server and shown that it can handle over 1.6 million requests per second with an average response time of 2ms. Furthermore, the authors in [35] present an extensible SDN control system where the authors implement a controller using the Glasgow and Haskell Compiler and determine that their solution can serve up to 5000 switches with a single controller of 46 cores. They were able to achieve a throughput of 14 million flows per second with a latency of 200 $\mu s$. One controller might serve well in many scenarios but sometimes we need more than one to provide redundancy, load balancing, geographically local footprint or scalability [36].

Introducing multiple controllers brings up many questions and the authors in [37] look at the question of how to separate areas in its domains and which controllers should be responsible for those areas. Research from distributed computing is used to apply local algorithms to better understand how to keep controller events within their own domain. To do so, it is suggested that link assignment from controller to switches can be solved using semi-matching problem. Here, given that there are many domains and many controllers, the authors proposed previous greedy algorithms to link each domain to a controller. Furthermore, the authors suggest the flat or hierarchical layout of the controllers, as shown in Figure 2.4. The flat model performs the the same way as the open shortest path first routing in relation to state and events in a network. In the hierarchical model, the top-most layer has complete knowledge of the network and only synchronizes with the two other layers underneath and the layer two levels underneath the top level synchronize with layers underneath it. The hierarchical model uses local algorithms to link its controllers to domains and results in a only locally optimal solution. Here, the suggested solution does not consider how big the domains are and could result in specific areas having a lot more controller traffic than others. The solutions could also include the different type of controllers where each controller can only handle a certain amount of traffic.



(a) Flat View



(b) Hierarchical View

**Figure 2.4:** Distributed controller views: (a) every area controller has a global view and synchronizes with another and (b) the root controller has a global view only and synchronizes down to lower levels.

### 2.3.6 Optimal Controller Placement

Similar to connecting controllers to switches in the previous paper, this section reviews the controller placement methods that are researched so far. The controller placement problem for SDN networks was first introduced in 2012 [38]. The authors present a method of placing controllers in a network and connecting switches with controllers that is based on k-median and k-center algorithms. These algorithms use a metric that determines which switches belong to which controller. The metric that is used is the latency from possible controller placement areas to the switches. Both, k-median and k-center are used to determine which of the algorithms perform better for different topologies. The algorithm assigns switches to a controller by clustering a set of switches and placing a controller to support such switches. The results are dependent on how the switches are laid out on a topology. If all the switches are equally the same latency (distance) away, then the clustering may have inconsistent results because it is not able to group all the switches properly. The two inputs that are used in the clustering algorithms are: "$K$" and cluster initialization. The input parameter $K$ instructs the algorithm that there must be exactly $K$ groups formed. The number of controllers installed in the network is the value of $K$. The initialization parameter is important because for the same $K$, if different initialization values are used, then different final clustering is formed [39]. This algorithm runs by trial and error because for the same topology, the algorithm has to be run multiple times to find the best clustering. This is a limitation because we believe that the optimal number of controllers and their locations must be found for each problem. A better method compared to clustering is the exact optimal placement of controllers. Furthermore, more than one parameter (latency) can be considered.

The authors in [40] propose a greedy approach to the placement of controllers in a network by maximizing the reliability of an SDN network. The probability of failure of each network component is known and on each candidate controller location, the shortest paths to all switches is found. A controller location is picked based on the minimum failure from the components and the shortest paths.

The authors in [41] propose a more complete solution by dynamically provisioning controllers. It is a framework that automatically adapts the number of controllers and links that are active while keeping the state of the network in consideration. The framework collects statistics, sets up flows, synchronizes between controllers and reassigns switches to controllers in real time. The costs of all of those tasks are kept to find the optimal solution. An optimal solution is found by greedy knapsack algorithm and simulated annealing. Since the framework always adapts to current network traffic, the greedy solution could possibly assign switches to controllers with the existing configuration and thus producing overhead. The simulated annealing algorithm takes into consideration the current state of the assignment and does not reassign the same configuration. In a network, it is realistic to have different connection capacities between network elements but the authors have neglected this fact. Therefore, different link capacities should be included in the overall optimization and a method for optimal placements of controllers in the network. The framework only answers the question of the optimal switch to controller assignment while taking the current traffic patterns and the overhead the framework creates into consideration. Here, the authors assume that there are more controllers present and it is possible to have idle controllers in the network.

## 2.4 Concluding Remarks

The literature review shows that the optimal controller placements only involve one or two input parameters. In the case of placing controllers using clustering, only latency is used to determine controller placement locations. Furthermore, a greedy approach that improves reliability minimizes the failure probability while keeping shortest distance between installed controller and switches. A framework that automatically assigns links to switches from the controllers assumes that the controllers are already placed in an SDN network.

In the next chapter, we will find optimal solutions models that place controllers on a new network or expand an existing SDN network. Besides minimizing the cost of the network, various constrains like: link bandwidth between switches and controllers, controller and link inventory, flow-setup latency and controller to controller connectivity will also be considered.

# Chapter 3

# Optimal Placement of Controllers in SDN

On a network of many switches, finding the best location to install a single controller is not an easy task. The same question is relevant when finding the best placement locations for multiple controllers. This chapter introduces the formulation of a mathematical model that simultaneously determines the optimal number, location, and type of controller(s) as well as the interconnections between all the network elements.

The chapter starts with an introduction on the general optimization problems. Then the controller placement problem is introduced and formulated.

## 3.1 General Optimization Problems

Optimization problems are used in many industries because the aim is to improve an objective [42]. Optimization is widely used in the networking field for various purposes: channel assignments, task scheduling in cloud computing, cellular network planning and many other areas [43]. Such a problem has the form $(f, \beta, I)$, where

- $f$ is the objective function that must be *min*imized or *max*imized along with the *decision variables*,
- $\beta$ is a set of constraints that involves the *decision variables* which, in turn, affects the size of the search space and
- $I$ is the input to the problem that defines the search space.

The objective function and constrains indicate whether the problem is linear or non-linear. Section 3.2.2 introduces a binary integer problem for placing controllers in a SDN network.

## 3.2 Planning Model

### 3.2.1 Known Information

In order to formulate the mathematical model, we assume the following information is known:

- The location of all the switches in the network. For each switches, the traffic that is generated to the controllers is also known.
- The bandwidth available for each link type to connect the switches and the controllers.
- The characteristics of the different types of controllers that may be placed in a network. Each type of controller has a cost (dollar), a number of physical ports available, a maximum number of requests it can handle per second and the number of controllers available for each type.
- The maximum link setup latency allowed for switch to controller communications.

### 3.2.2 Problem Formulation

This section formulates the binary integer problem. A network topology is provided by an undirected graph $G = (S, E)$ where $S$ is a set of switches and $E$ is a set of edges. The following notation is composed of sets, decision variables and constraints:

**Sets**

- $S$, a set of switches that are present in the network, $S = \{s_1, s_2, s_3 \dots\}$
  - $\sigma^s$, the number of packets that do not match on the switch ($s \in S$) table and that must be sent to the future connected controller.
- $C$, a set of controller types that can be installed, $C = \{c_1, c_2, c_3 \dots\}$.
  - $\alpha^c$ is the number of ports available for the controller type $c \in C$.
  - $\mu^c$ is the processing power in packets per second for a controller of type $c \in C$.

  - $\kappa^c$ is the price in dollars of the controller type $c \in C$.
  - $\varphi^c$ is the number of controllers available of type $c \in C$.

- $L$, a set of possible link types that can be used to connect controllers and switches, $L = \{l_1, l_2, l_3 \dots\}$.
  - $\omega^l$ is the bandwidth in Mbps of link type $l \in L$.
  - $\phi^l$ is the price in dollars per meter of link type $l \in L$.
- $P$, the set of possible locations to install the controllers.

**Constants**

- $\beta$, is the packet size for each packet that is sent to the controller in bytes.
- $\gamma$, is the maximum delay allowed in the network for flow-setup.
- $\delta$, is the average time in milliseconds taken to process a packet by any controller.
- $t$, is the speed of light.

**Functions**

- `dist(a,b)` is a function that calculates the distance between point $a$ and point $b$. The value of $a$ and $b$ are in the form $a = (x, y)$ and $b = (x, y)$.
- `f(a)` is a function where $a$ is a value in Mbps or Gbps and that converts to bytes per second.
- `g(a,b)` is a function that takes the input parameter $a$ as the number of packets per second, $b$ the packet length in bytes and converts it to bytes per second.
- `Prop(...)` is the propagation delay (see Section 3.2.2).
- `Tran(...)` is the transmission delay (see Section 3.2.2).
- `Proc(...)` is the processing delay (see Section 3.2.2).

**Decision Variables**

- $x_{cp} = \begin{cases} 1, & \text{if controller of type } c \in C \text{ is installed on location } p \in P; \\ 0, & \text{otherwise.} \end{cases}$

- $v_{sp}^{l} = \begin{cases} 1, & \text{if link of type } l \in L \text{ and switch } s \in S \text{ is installed on location } p \in P; \\ 0, & \text{otherwise.} \end{cases}$

- $z_{pq}^{l} = \begin{cases} 1, & \text{if location } p \in P \text{ is connected to location } q \in P \text{ of link type } l \in L; \\ 0, & \text{otherwise.} \end{cases}$

**Objective Function**

The objective function of the planning model is to minimize the total cost of placing controllers and links. The total cost of the optimization model is the cost of placing the controllers, the cost of linking the controllers to the switches and the cost of linking controllers together. The cost of placing controllers is $C_c(x)$, the cost of placing links between controllers and switches is $C_l(v)$ and the cost of linking controllers together is $C_t(z)$. The variable $x$ is a matrix that holds boolean values for each controller of type $c \in C$ that is placed at possible location $p \in P$. The variable $v$ is a matrix that holds the boolean values for each link type $l \in L$ where each possible controller location $p \in P$ is connected to a switch $s \in S$. Finally, $z$ is also a matrix that holds boolean values of link type $l \in L$ that connects a controller $p \in P$ and another controller $q \in P$.

$$C_c(x) = \sum_{c \in C} \kappa^c \sum_{p \in P} x_{cp} \tag{3.1}$$

$$C_l(v) = \sum_{l \in L} \phi^l \sum_{s \in S} \sum_{p \in P} \texttt{dist}(s,p) v_{sp}^{l} \tag{3.2}$$

$$C_t(z) = \sum_{l \in L} \phi^l \sum_{q \in P} \sum_{\substack{p \in P \\ q < p}} \texttt{dist}(q,p) z_{pq}^{l} \tag{3.3}$$

The planning problem can be modelled with the following function:

$$\texttt{Minimize } \left(C_c(x) + C_l(v) + C_t(z)\right)$$

**Formulation Constrains**

- Enforce uniqueness constraint for controllers.

$$\sum_{c \in C} x_{cp} \leq 1 \qquad (p \in P) \tag{3.4}$$

- Make sure that the number of connected switches and controllers to one specific controller is smaller than the number of ports on the controller.

$$\sum_{q \in P} \sum_{l \in L} \left(z^l_{pq} + z^l_{qp}\right) + \sum_{s \in S} \sum_{l \in L} v^l_{sp} \leq \sum_{c \in C} \alpha^c x_{cp} \qquad (p \in P) \tag{3.5}$$

- Make sure that exactly one link (of any type $l \in L$) is installed between a given switch and the controllers.

$$\sum_{l \in L} \sum_{p \in P} v^l_{sp} = 1 \qquad (s \in S) \tag{3.6}$$

- Make sure that the number of packets each switch sends can be processed by the controller.

$$\sum_{l \in L} \sum_{s \in S} \sigma^s v^l_{sp} \leq \sum_{c \in C} \mu^c x_{cp} \qquad (p \in P) \tag{3.7}$$

- Make sure that the inventory of each controller is not exceeded.

$$\sum_{p \in P} x_{cp} \leq \varphi^c \qquad (c \in C) \tag{3.8}$$

- Make sure that the link that is chosen between the controller and the switch can handle the bandwidth needed by the switch.

$$g\left(\sigma^s, \beta\right) \leq \sum_{l \in L} f\left(\omega^l\right) v^l_{sp} \qquad (s \in S, p \in P) \tag{3.9}$$

- Keep the round trip flow-setup latencies for unmatched flows in each of the switches below or equal to $\gamma$.

$$2\ \texttt{Tran}(v) + \sum_{c \in C} \left(2\ \texttt{Prop}(x, v) + \texttt{Proc}(x)\right) \leq \gamma \tag{3.10}$$

- Connect the controllers together using the full mesh topology.

$$\sum_{c\in C} x_{cq} + \sum_{c\in C} x_{cp} \leq \sum_{l\in L} z^l_{pq} + 1 \qquad (q < p, q \in P, p \in P) \tag{3.11}$$

- Ensure that valid values are assigned to the variables representing controllers and links.

$$x_{cp} \in \{0, 1\} \qquad (c \in C, p \in P) \tag{3.12}$$

$$v^l_{sp} \in \{0, 1\} \qquad (l \in L, s \in S, p \in P) \tag{3.13}$$

**Inner-Controller Topology Connectivity**

Controllers can be physically connected using various topologies such as a tree, ring, and full mesh. In this thesis, the full mesh topology is used to connect controllers together. The authors in [44] provide the formulations and mathematical proofs for each of the topologies.

To connect any of the topologies, there must be a decision variable that must be added to the model. This decision variable makes it possible for our objective to include the extra links that are needed between controllers. The variable is $z^l_{pq}$, which indicates if controller placement location $p$ is connected to another controller location $q$ with a link type $l$. Equation 3.11 forms a constraint that connects controllers together in full mesh topology.

The equation has a restriction that the index of the controller installed at location $p$ must always be greater than the index of the controller installed at location $q$. This ensures that only one directional link is placed from controller $p$ to controller $q$. The topology equation only places one way connectivity ($p$ to $q$). In Equation 3.5, when we count the number of ports used for the controllers, it is necessary to also include the topology link in the opposite direction ($q$ to $p$). The reason behind this is because only one link is placed between controllers and it is either $z^l_{pq}$ or $z^l_{qp}$ and we do not know which variable is set.Therefore, the addition of $z^l_{pq}$ and $z^l_{qp}$ is needed to make sure a link between any two controllers is always counted.

### Flow-Setup Latencies

Flow setup is initiated on a switch only if a flow is not found on the switch table and that switch forwards the flow to the controller to determine where to send the flow. Depending on the configuration of the OpenFlow, either the whole packet will be sent or $\beta$ amount of bytes of the packet is sent to a controller. This thesis works under the assumption that only $\beta$ amount of bytes is sent. The reasoning behind this is that there is no need to send full 1500 packet bytes, since that would result in a lot of overhead. The latencies for flow setup are calculated using the propagation, controller processing and transmission delays. We believe that the queuing delay should also be added to form a full end to end delay calculation. Adding queuing mechanisms is out of the scope of this thesis.

Propagation is the time taken to transmit a signal from a source to a destination. This depends on the medium used. Propagation is calculated as $\frac{\texttt{distance}}{\texttt{propagation speed}}$. Depending on the medium that is used, the propagation speed varies. The propagation speed of wireless communications is the speed of light. For copper wires, the speed varies from $0.59t$ to $0.77t$. In the model, we use $0.59t$ for the speed of copper wire.

Controller Processing is the time taken to process the request in a controller and to make a decision of what should be done with the new flow. Maybe a new rule would have to be installed back into the switch. This involves the time taken to run the algorithm that generates the rules. These times depend upon the processing capacity of the controller. The processing of each packet is assumed to take $\delta$ milliseconds.

Transmission delay is the time taken for a process to send the information to the wire. This includes the time taken at each layer of the TCP/IP down until the bit level layer. This depends on the link speed that is used and the packet size that is to be sent to the controller. The formula to calculate transmission is $\frac{\texttt{packet size (bytes)}}{\texttt{link speed (bytes/sec)}}$.

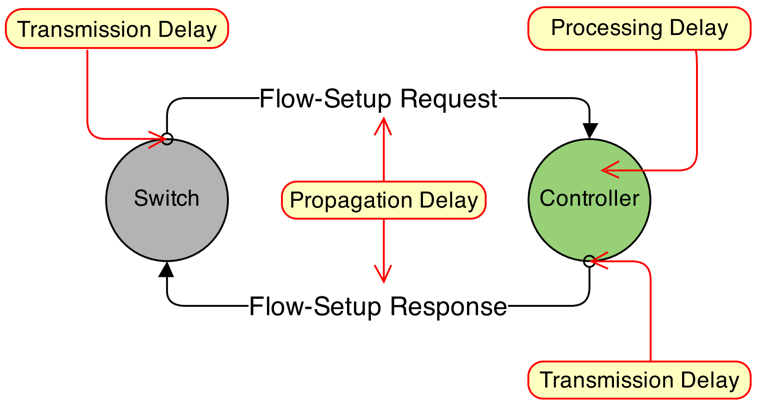


**Figure 3.1:** The different delays that are calculated in flow setup latency.

Figure 3.1 shows the delays that need to be included into the calculation of flow-setup latency. Each of the delays can be calculated using the following formulas:

Propagation Delay - $\texttt{Prop}(x, v)$:

$$\left(\frac{\texttt{dist}(s,p)}{0.59t}\right) x_{cp} \tag{3.14}$$

Transmission Delay - $\texttt{Tran}(v)$:

$$\left(\frac{\beta}{\omega^l}\right) v^l_{sp} \tag{3.15}$$

Processing Delay - $\texttt{Proc}(x)$:

$$(\delta)\, x_{cp} \tag{3.16}$$

The planning model presented in this section only works when no controllers are already installed. This is often referred to as green field deployment. However, in most situations, an existing infrastructure will already be in place. As a result, it is important to consider existing equipment that is already installed into the planning process. The next section will introduce an expansion model that is used to alter, if necessary, existing solutions. In this model, new decision variables, cost variables, constrains and cost function are presented.

## 3.3 Expansion Model

In the case if information technology had no limitations on its resources or an unlimited budget was provided to build a large network, it is possible that a large enough network could be built and it would not require any changes. However, current networks have limitations in resources or budgets that make it an ongoing effort to maintain good services at the lowest possible price. This section will introduce an expansion model for placing controllers in an SDN network. In fact, the expansion model is a generalization of the planning model discussed in the previous section. As a result, this new model can be used to plan a brand new SDN network or expand an existing network. The first section introduces the problem with the changed objective and new constraints. The second section formulates the mathematical model.

### 3.3.1 Known Information

The expansion model also needs all the information that was needed by the placement model as mentioned in Section 3.2.1. In addition, the following information is also needed:

- The existing SDN topology. This includes the location and the type of controllers that are installed as well as the topology (links) between the various network elements.
- The cost of removing an existing link between switches and controllers.
- The cost of removing an existing link between controllers.
- The cost of reallocating controllers in the network.

### 3.3.2 Problem Formulation

In this model, we will be introducing a new cost function with new cost variables and decision variables.

#### Sets

In addition to the sets mentioned in the planning phase (see Section 3.2.2), the following are added:

- $\bar{\kappa}^c$ is the cost in dollars to remove a controller of type $c \in C$.
- $\bar{\phi}^l$ is the cost in dollars to remove a link per meter of link type $l \in L$.

**Decision Variables**

- $\bar{x}_{cp} = \begin{cases} 1, & \text{if controller of type } c \in C \text{ is already installed in location } p \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $\bar{v}^l_{sp} = \begin{cases} 1, & \text{if link of type } l \in L \text{ and switch } s \in S \text{ is already installed on location } p \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $\bar{z}^l_{pq} = \begin{cases} 1, & \text{if location } p \in P \text{ is already installed to location } q \in P \text{ of link type } l \in L; \\ 0, & \text{otherwise.} \end{cases}$
- $Rv^l_{sp} = \begin{cases} 1, & \text{if existing installation link } l \in L \text{ is removed from } s \in S \text{ and } p \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $Kv^l_{sp} = \begin{cases} 1, & \text{if a new link } l \in L \text{ is connected from switch } s \in S \text{ and placement } p \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $Rz^l_{pq} = \begin{cases} 1, & \text{if existing toplology link } l \in L \text{ is removed from } p \in P \text{ and } q \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $Kz^l_{pq} = \begin{cases} 1, & \text{if a new link } l \in L \text{ is connects installed controllers } p \in P \text{ and } q \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $Rx_{cp} = \begin{cases} 1, & \text{if controller } c \in C \text{ is removed from location } p \in P; \\ 0, & \text{otherwise.} \end{cases}$
- $Kx_c =$ is the number of controllers of type $c \in C$ that will be installed.

**Objective Function**

Depending on the goal, the model will expand an existing placement problem or perform placements for the first time. The objective of the function is to minimize the cost of placing controllers on a network. The functions that minimize the cost are:

$C'_c(x)$ for placing controllers, $C'_l(v)$ for connecting controllers and switches together, and $C'_t(z)$ for connecting new or installed controllers together.

$$C'_c(x) = \sum_{c \in C} \left( \kappa^c \left( K x_c \right) + \bar{\kappa}^c \left( \sum_{p \in P} R x_{cp} \right) \right) \tag{3.17}$$

$$C'_l(v) = \sum_{l \in L} \left( \phi^l \left( \sum_{s \in S} \sum_{p \in P} \texttt{dist}(s,p)(K v^l_{sp}) \right) + \bar{\phi}^l \left( \sum_{s \in S} \sum_{p \in P} \texttt{dist}(s,p)(R v^l_{sp}) \right) \right) \tag{3.18}$$

$$C'_t(z) = \sum_{l \in L} \left( \phi^l \left( \sum_{q \in P} \sum_{\substack{p \in P \\ q<p}} \texttt{dist}(q,p) K z^l_{pq} \right) + \bar{\phi}^l \left( \sum_{q \in P} \sum_{\substack{p \in P \\ q<p}} \texttt{dist}(q,p) R z^l_{pq} \right) \right) \tag{3.19}$$

The expansion problem can be modelled with the following function:

$$\texttt{Minimize} \;\; \left( C'_c(x) + C'_l(v) + C'_t(z) \right)$$

**Formulation Constrains**

Based on the information from above, the following constrains are needed to formulate the expansion model:

- Planning or reallocating same type of controllers:

$$\sum_{p \in P} x_{cp} - K x_c \leq \sum_{p \in P} \bar{x}_{cp} \qquad (c \in C) \tag{3.20}$$

- Removing an existing controller:

$$R x_{cp} + x_{cp} \geq \bar{x}_{cp} \qquad (c \in C, p \in P) \tag{3.21}$$

- Planning a link between the switch and controller.

$$v^l_{sp} - K v^l_{sp} \leq \bar{v}^l_{sp} \qquad (l \in L, s \in S, p \in P) \tag{3.22}$$

- Removing an existing link between the switch and controller.

$$Rv^l_{sp} + v^l_{sp} \geq \bar{v}^l_{sp} \qquad (l \in L, s \in S, p \in P) \tag{3.23}$$

- Planning a link between controllers $p$ and $q$.

$$z^l_{pq} - Kz^l_{pq} \leq \bar{z}^l_{pq} \qquad (q \neq p, l \in L, q \in P, p \in P) \tag{3.24}$$

- Removing an existing link between controllers $p$ and $q$.

$$Rz^l_{pq} + z^l_{pq} \geq \bar{z}^l_{pq} \qquad (q \neq p, l \in L, q \in P, p \in P) \tag{3.25}$$

- Integrity constraints for $Kx_c$.

$$Kx_c \geq 0 \qquad (c \in C) \tag{3.26}$$

- Integrity constraint for $Rx_{cp}$.

$$Rx_{cp} \geq 0 \qquad (c \in C, p \in P) \tag{3.27}$$

- Integrity constraint for $Kv^l_{sp}$.

$$Kv^l_{sp} \geq 0 \qquad (l \in L, s \in S, p \in P) \tag{3.28}$$

- Integrity constraint for $Rv^l_{sp}$.

$$Rv^l_{sp} \geq 0 \qquad (l \in L, s \in S, p \in P) \tag{3.29}$$

- Integrity constraint for $Kz^l_{pq}$.

$$Kz^l_{pq} \geq 0 \qquad (q \neq p, l \in L, q \in P, p \in P) \tag{3.30}$$

- Integrity constraint for $Rz^l_{pq}$.

$$Rz^l_{pq} \geq 0 \qquad (q \neq p, l \in L, q \in P, p \in P) \tag{3.31}$$

### 3.3.3 Logical Constraints

The logical constraints used in the problem formulation (equations 3.20 to 3.25) allow the model for choosing between three better options when optimizing for the objective function. They take the form $x + a \geq z$ or $x - b \leq z$, where $x, a, b$ and $z$ are binary variables. Variables $a$ and $b$ are included in the cost functions such as $\min(P_1 a + P_2 b)$ where $P_1$ and $P_2$ are costs (dollars). The goal is to select either variable $a$ or $b$, or to not select one at all.

In the case of $x + a \geq z$, which is the exclusivity constraint, either $x$ or $a$ can be set to 1 when $z$ is 1. This is because $1 + 0 \geq 1$ and $0 + 1 \geq 1$ hold. If $a$ is set to 1, then price $P_1$ is added to the objective. If $a$ is not set then the constraint has no effect on the objective function. As for the constraint $x - b \leq z$, when $z$ is set to 1, only $x$ can be set to 1 and $b$ will always be forced to 0. If $z$ is set to 1, both $x$ and $b$ are either set to 1 or 0. We can see that if $z$ is set to 1, then $b$ is always 0, and $a$ may be 1. However, when $z$ is set to 0, then $a$ is always 0.

Equations 3.21, 3.23 and 3.25 check if a link that already exists can still be included. If for instance the variable $v^l_{sp}$ is forced to be 0 by a constraint from the planning model, then the remove variable is set to 1 ($Rv^l_{sp}$). The cost of removing such a link is included in the cost calculation of the optimal solution. If a link does not exist already, then the cost variable is not selected, thus not changing the price of the solution (the constraint is ignored).

The purpose of Equation 3.22 and 3.24 is to plan links. There are general constraints that consider if a link already exists or not. If a link already exists then its price is not added to the total cost of the solution. This is because the right hand side of the equations will be set to 1 and only the left most variable ($v^l_{sp}$) of the left side of the inequality can be set to 1. If a link does not exist, then both variables on the left hand side of the inequality are set to 1. This ensures that the price of the new link should be included in the calculations.

## 3.4 Concluding Remarks

In this chapter, we have formulated two models for placing controllers in a network. Depending on the goal, the first model is used to decide new placement locations for controllers and connect the network whereas the second model is a generalization of the planning problem and can be used to plan or expand a network. In the next chapter, we show how the models perform under different conditions and we analyze the results.

Chapter 4

# Results and Analysis for the Controller Placement

In this chapter, we show the collected results for the purpose of understanding the performance and limits of the models. First, the placement problem of controllers is solved with different topologies. Then, the expansion of already placed controllers is determined and results are given. The chapter starts with simple examples of the models and then continues by examining more complex optimized scenarios.

## 4.1 Placement Problem: Results and Analysis

Placement of controllers in a new network is not a simple task. One difficult aspect of it is the gathering of information about the topology that is to be placed using the planning model. The information (input parameters to the model) is essential to the placement of controllers. For example, the location of switches and possible controller locations needs to be known, as well as the inventory information that is available to place the controllers. Subsection 4.1.2 goes into detail by showing a simple example of the planning problem. We begin with the input parameters and then go into detail about each of the constrains related to the example that is solved.

### 4.1.1 Methodology

In our implementation, there are many steps that we have to take in order to find an optimal solution. All the steps are summarized in Figure 4.1. First, the topology is generated from the input such as those provided in Tables 4.1, 4.2 and 4.3. This forms the set of switches ($S$) and the set of possible controller placement locations ($P$).

Next, the linear problem model file is generated using the input information gathered so far and including the information that is in Table 4.3. At this time, all the information is available for the model to be generated. Using Python programming language, the model is generated. The objective functions and constraints are expanded using software and are saved to a linear program model file (`Generate-Model()`). A solver optimizes the problem (`Cplex-Optimize()`) using the linear program as its input. Once the solver finds an optimal solution, the result is saved in a simple text file. Finally, the result file is read and interpreted and a plot is generated (`Analyze-Plot()`).

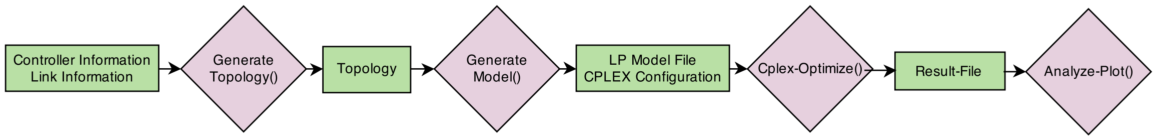


**Figure 4.1:** Steps to find the optimal solution for placing controllers on a network.

The location input to the model is generated randomly for every location variable. The placement and the expansion examples locations are generated within 100 meters by 100 meters by randomly generating the values for the switches and possible controllers locations. A location of a switch or possible location is a point form ($x$,$y$) in a graph. A random integer between 0 and 100 (exclusive) is assigned to $x$ and $y$. For the result analysis input information, an area of 1000 meters by 1000 meters is used instead. With a larger area, the result graphs display better because we generate many switches and controller placement locations. Each switch that sends packets to its future connected controller ($\sigma^s$) is assigned a random integer value between 0 and 999 (exclusive).

The model is generated using the command line scripts:

```
python cli.py --in=./input --out=./output --cplex --settings=
    simulation.txt --confsection=placement1
```

The optimization input information is stored in configuration files and the location of the configuration files is specified using the `--in` parameter. The specific configuration file is set with the `--settings`. Since each configuration file contains multiple topologies a topology is selected by specifying the `--confsection` instruction. The `--cplex` instruction tells the script to generate the CPLEX linear program model. After CPLEX solves the problems, we analyze the model and graph it using the same command as above but instead of instructing the script to generate the model, we instruct the script to plot and analyze the model by replacing the argument `--cplex` to `--plot` (see Appendix C). Since there will be many scenarios to optimize, we will schedule our optimizations on a cluster of computers using the Sun Grid System. Once the scheduler runs, we follow the same steps mentioned earlier to optimize each problem. The following are files that are generated by end of each problem:

- `cpp_com_file_run_planning.com`, which is a bash file and is what the problem is scheduled with in the cluster,
- `cpp_problem_run_planning.lp`, which is the linear program model file that is read and optimized by CPLEX,
- `cpp_solution_run_planning.txt`, which is the output to the optimization, and
- `report_run_planning_img.pdf` is the plot of the solution.

To graph the solution, Matplotlib library is used with Python. The graph itself automatically determines the minimum and maximum $x$ and $y$ coordinates that are used as the graph dimensions. Set $S$ and $P$ are initially plotted. Once the solution is found, the controllers are plotted and the links that have been selected between controller and switches. Furthermore, if multiple controllers exist, then the links between controllers are also plotted.

### 4.1.2 Detailed Example

The simple example in this section shows the understanding of the planning model before the optimization results are analyzed. In this example, a network composed of 7 switches is solved by finding the optimal location(s) of controller(s) placement while considering the controller processing, the bandwidth between switches and controllers, flow setup latency, link and controller inventory and controller to controller connectivity. Once the result is obtained, it will be analyzed and the final conclusion is made by showing the final cost and the optimal placement variables.

The placement model in the first section of Chapter 3 is used with the inputs shown in Table 4.1 and Table 4.2.

In this simple scenario, we have two types of controllers as shown in Table 4.1. The first type is the cheapest and costs \$10, comes with five physical ports and can process

**Table 4.1:** The type of controllers avaliable.

| | Type 1 | Type 2 |
|---|---|---|
| **Cost** ($\kappa^c$) | \$10 | \$50 |
| **Ports** ($\alpha^c$) | 4 | 8 |
| **Processing** ($\mu^c$) | 100 | 300 |
| # **Controllers** ($\varphi^c$) | 2 | 1 |

**Table 4.2:** The type of links avaliable.

| | Type 1 | Type 2 |
|---|---|---|
| **Cost/meter** ($\phi^l$) | \$0.05 | \$1.99 |
| **Bandwidth** ($\omega^l$) | 1 Mbps | 10 Mbps |

100 packets per second. In the inventory, there exist only two of these controllers. The second type costs \$50, comes with eight physical ports and can process 300 packets per second. There is only one controller available in the inventory for type two. In total, the three controllers cost \$70, have 18 ports and can process 500 packets per second.

The link information is shown in Table 4.2. Two link types exist with bandwidth of 1 Mbps and 10 Mbps, respectively. The price of these links is shown in dollars per meter. The first link costs \$0.05 per meter and the second costs \$1.99 per meter. There are no restrictions on how many meters of each type of link are used and it is assumed that unlimited cabling is available.

For the simple example, the initial topology includes seven switches and five possible placement locations for the controllers. We assume each switch sends a certain number of packets per second ($\sigma^s$) to the controller to determine what to do with the unknown packets. Figure 4.2 shows the location of all the switches that are already installed in the network (marked with blue dots) and the potential locations to install the controllers (marked with $\times$ symbol). The number on top of the switches is the

**Table 4.3:** Toplology and optimization input parameters for the example.

| | |
|---|---|
| **Maximum Latency** | 250 ms |
| **Packet Size** | 1,500 Bytes |
| **CPLEX Restrictions** | Single Thread Run |
| **Simulation Area** | 100x100 meters |
| **Link Available** | Unlimited |
| **Controllers Available** | 3 |
| **Number of Switches** | 7 (random) |
| **Maximum Placements** | 5 (random) |
| **Switch to Controller** ($\sigma^s$) | $[290, 11, 12, 16, 13, 34, 15]$ |
| **Maximum Optimization Duration** | 10,8000 Seconds |

number of packets that the switch sends to the future connected controller ($\sigma^s$). The area that is used in this scenario is 10,000 meters squared.

By visual observation of Figure 4.2, we can tell that switches $S_2$ and $S_5$ are close to placement location $P_4$. Also, the distance between $P_4$ and $S_6$ is 34 meters whereas the distance between $P_3$ and $S_6$ is 41 meters. Therefore, it is very likely that $S_6$ connects to $P_4$. For switches $S_1$ and $S_4$, controller placement location $P_2$ is the closest. As for the last two switches, $S_3$ and $S_7$, the closest controller placement location may be $P_5$. The possible placement location $P_3$ and $P_1$ may not have any controllers. To better understand the constraints of the planning model presented in this section, the example is applied to each of the constrains mentioned in the planning model of Chapter 3.

## Controller Processing Capacity

Equation 3.7 states that the incoming requests from the switches must be less than the processing capacity of the controller. For instance, switch $S_1$ sends 290 packets per second to the future connected controller. The only controller that can process this many requests is the controller of type two that has a capacity of 300 packets per second. Therefore, the type two controller is assigned on location $P_2$ and $S_1$ is

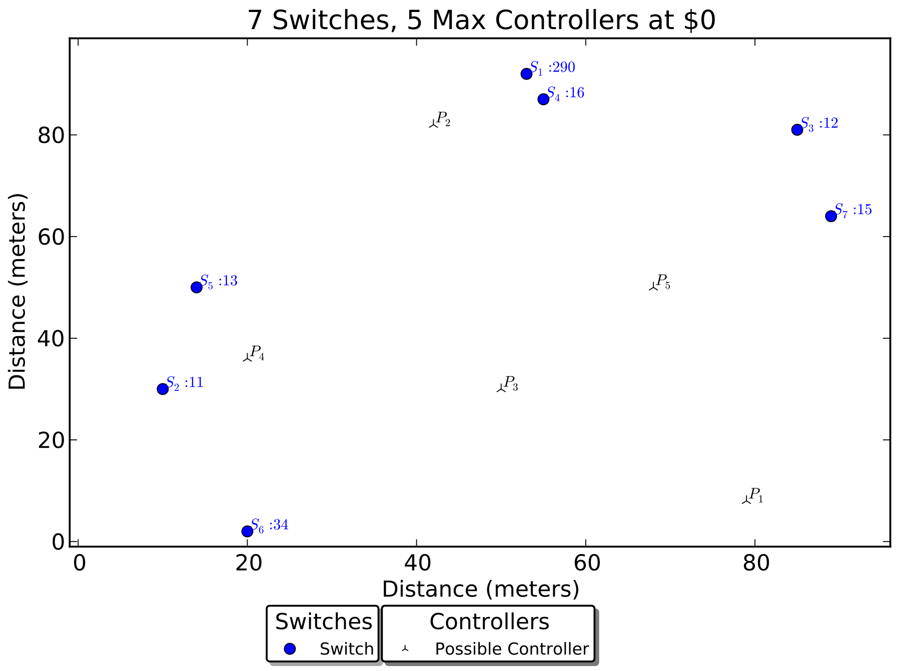


**Figure 4.2:** The location of the switches and the potential controller locations for the placement example.

assigned to it. The variables for the controller placement ($x_{cp}$) and link between switch and controller ($v^{l}_{sp}$) are set. Variable $x_{22} = 1$ and $v^{?}_{12} = 1$. Note that the question mark indicates that the link type is currently unknown. The installed controller can process any other switch that sends at most ten packets per second. However, all the other switches have more than 11 packets per second to send to the controller.

The total processing that is needed for the network by the switches is 391 packets per second. Since $S_1$ is connected to a controller already, the remaining processing capacity is 101 packets per second for the whole network. That is low enough to be covered by a single type two controller. However, other constraints may fail that will force more controllers to be placed in the network.

### Controller Port Availability

Equation 3.5 checks for the port availability when connecting switches to installed locations of controllers and other controllers that connect to form the full mesh topology. We know that a controller is already placed in location $P_1$ and this means that only one port is used to connect a switch. This constraint passes for the installed controller on $P_1$ because seven ports remain available. One reason why there is more than one controller installed is because we have more switches compared to the available number of ports on the type two controller.

### One Link Connecting Controller and Switches

Equation 3.6 ensures that every switch is connected to at most one controller. So far, only switch $S_1$ is connected. Since we know that $P_2$ can only process one switch, others remain not allocated to a controller. Therefore, a controller is installed in location $P_4$. Switches $5, 6$, and 2 can be served by this controller because their distance is the shortest compared to other possible controller placements. This sets the variable $x_{cp}$ to $x_{14} = 1$ because a controller of type one is sufficient to handle these switches. Similarly, link placement sets variables $v_{sp}^{l}$ to $v_{54}^{?} = 1, v_{64}^{?} = 1$, and $v_{24}^{?} = 1$. The question mark indicates that the link type is not known. Connecting these three switches to the controller $P_4$ uses three ports on the controller. In addition, one port on each controller is used to connect $P_4$ to $P_2$. In total, four ports are currently used.

Switches $3, 4$, and 7 are not connected yet. Equation 3.5 only allows one more switch to be connected to the controller installed at location $P_4$ because there is only one physical port left. This means that a new controller must be installed somewhere close to the two switches that are disconnected. A new controller is placed and the optimal placement is on $P_5$. Any other available possible controller location would make the cost of connecting switches $S_3$ and $S_7$ more expensive due to the fact that the distance increases between the switch and controller. This is not always the case because controller to controller connectivity would also increase the cost function if $P_5$ was too far away. A controller is installed at location $P_5$ for switches $S_3$ and $S_7$ to be connected. Since these switches only require a controller with processing capacity of 27 packets per second, controller of type one is picked to minimize the cost. The variables that are set include: $x_{cp}$ to $x_{15} = 1$ and $v_{35}^{?} = 1$ and $v_{75}^{?} = 1$ for $v_{sp}^{l}$. This leaves switch $S_4$ not connected anywhere. The distance to $P_1$ is a lot closer than $P_5$ but the processing power of the controller of type two in $P_1$ is under limit of what

$S_4$ requires (10 vs 16). Therefore, $S_4$ must be connected to $P_5$. The variable set for connecting $S_4$ to $P_5$ is $v^?_{45} = 1$.

### Controller Inventory Limits

Equation 3.8 limits how many of each controller type can be installed. In the final solution, the solver will make sure each controller type is not used more than the number of controllers available as indicated in the controller input information in Table 4.1. In this case, two controllers of type one and one controller of type two are installed in the network.

### Link Bandwidth Between Switch and Controller

Until now, we have only shown that links must exist between switch and controllers. We could not specify what link is to be installed. To determine the correct link type to use, a link must be able to transmit the packets that a switch sends to the controller (see Equation 3.9). The function $g(\sigma^s, \beta)$ determines the needed bandwidth for a switch to communicate with a controller, where $\beta$ is the packet size in bytes. For this example, $\beta$ is 1,500 bytes (see Table 4.3). For the switch $S_1$, 290 packets per second are sent to the controller ($\sigma^1$). The total link bandwidth required for switch $S_1$ is calculated as follow.

$$g(\sigma^1, 150) = 290 * 1500 = 435,000 \text{ bytes per second.}$$

Links from Table 4.2 are in Mbps and need to be converted to bytes per second. Function $f(\omega^l)$ converts between the two units for link type ($l \in L$). Link types one and two are converted as follows.

$$f(\omega^1) = f(1\text{Mbit}) = \frac{1}{8} * 1,000,000 = 125,000 \text{ bytes per second, and}$$
$$f(\omega^2) = f(10\text{Mbit}) = \frac{10}{8} * 1,000,000 = 1,250,000 \text{ bytes per second.}$$

The bandwidth that the packets from $S_1$ generate to the installed controller ($P_1$)

cannot be satisfied by a link of type one because the link bandwidth is lower (435,000 bytes per second vs 125,000 bytes per second). Therefore, the link assignment ($v^l_{sp}$) from $S_1$ to $P_1$ is of link type two ($v^2_{12}$). The rest of the connections between the installed controllers and the switches can be handled by links of type one because the next highest outgoing packet rate from the switches is 34 packets per second (47,600 bytes per second).

### Controller Connectivity

Equation 3.11 ensures that the controllers are connected together. The equation only looks at possible placement locations where controllers have been installed ($x_{cp}$). If the equation is expanded, it starts from the lowest index and moves on to connect controllers that have a lower index then its own. At possible controller location one ($P_1$), a controller is not installed and since this is the first controller, nothing happens. At the second location ($P_2$), there exists a controller but it is the only installed controller. So far, there is no work to do because there is nothing to connect $P_2$ to. The next installed controller is on possible location five ($P_5$). Connections to the previous installed controllers are made. This means that it connects $P_4$ to $P_2$ and sets the variable $z^?_{pq}$ to $z^?_{42}$. The next installed controller in order is $P_5$ and this controller is connected with installed controllers on locations 4 and 2. Therefore, the next variables for topology connectivity are $z^?_{52}$ and $z^?_{54}$. The fact that we only consider minimal traffic between controllers, the lowest link type is used. The variables that are set to connect the controllers are $z^l_{42}$, $z^1_{52}$ and $z^1_{54}$.

In summary, the variables that are selected are:

- Controller Placement ($x_{cp}$): $x_{22}, x_{15}, x_{14}$
- Switch and Controller Connectivity ($v^l_{sp}$): $v^2_{12}, v^1_{75}, v^1_{54}, v^1_{64}, v^1_{24}, v^1_{35}, v^1_{45}$
- Full Mesh Controller Topology ($z^l_{pq}$): $z^1_{42}, z^1_{52}, z^1_{54}$

### Cost of the Solution

The total cost to the above scenario is the cost for each of variables that have been chosen from $x_{cp}$, $v^l_{sp}$ and $z^l_{pq}$. All the costs added together come to \$114 as shown in Table 4.4. The selected variables are marked with a box on the table. Since this is a small example, we are able to show partial cost function to illustrate the selection of the variables that make up the cost.

**Table 4.4:** The cost formulation of the planning example.

| Controllers($x_{cp}$) | Links($v_{lsp}$) | Topology($z_{lpq}$) |
|---|---|---|
| $\$10x_{11}$ | $\$4.397v_{111}$ | $\$4.137z_{121}$ |
| $\$10x_{12}$ | $\$0.743v_{112}$ | $\$1.820z_{131}$ |
| $\$10x_{13}$ | $\$3.104v_{113}$ | $\$3.265z_{141}$ |
| $\boxed{\$10x_{14}}$ | $\$3.250v_{114}$ | $\$2.171z_{151}$ |
| $\boxed{\$10x_{15}}$ | $\$2.230v_{115}$ | $\$2.631z_{132}$ |
| $\$50x_{21}$ | $\$3.621v_{121}$ | $\boxed{\$2.550z_{142}}$ |
| $\boxed{\$50x_{22}}$ | $\$3.053v_{122}$ | $\boxed{\$2.062z_{152}}$ |
| $\$50x_{23}$ | $\$2.000v_{123}$ | $\$1.530z_{143}$ |
| $\$50x_{24}$ | $\boxed{\$0.583v_{124}}$ | $\$1.345z_{153}$ |
| $\$50x_{25}$ | $\$3.068v_{125}$ | $\boxed{\$2.500z_{154}}$ |
| | $\$3.662v_{131}$ | $\$164.642z_{221}$ |
| | $\$2.151v_{132}$ | $\$72.437z_{231}$ |
| | $\$3.093v_{133}$ | $\$129.961z_{241}$ |
| | $\$3.953v_{134}$ | $\$86.399z_{251}$ |
| | $\boxed{\$1.768v_{135}}$ | $\$104.697z_{232}$ |
| | ...4 Observations cut off... | |
| | $\boxed{\$1.961v_{145}}$ | $\$99.500z_{254}$ |
| | $\$3.869v_{151}$ | |
| | $\$2.126v_{152}$ | |
| | $\$2.059v_{153}$ | |
| | $\boxed{\$0.762v_{154}}$ | |
| | ...5 Observations cut off... | |
| | $\boxed{\$1.700v_{164}}$ | |
| | ...5 Observations cut off... | |
| | $\boxed{\$1.262v_{175}}$ | |
| | $\$174.984v_{211}$ | |
| | $\boxed{\$29.583v_{212}}$ | |
| | $\$123.524v_{213}$ | |
| | ...31 Observations cut off... | |
| | $\$50.225v_{275}$ | |
| Total Cost | | \$114 |

### CPLEX Optimization

As the problem size increases, it becomes very difficult (not to say impossible) to solve the problems manually and that is why a solver is needed. The optimization for this scenario is run on IBM's ILOG CPLEX Optimizer version 12.5. The optimizer runs on a single thread. The process that is shown in Figure 4.1 is run on a computer that has the solver installed.

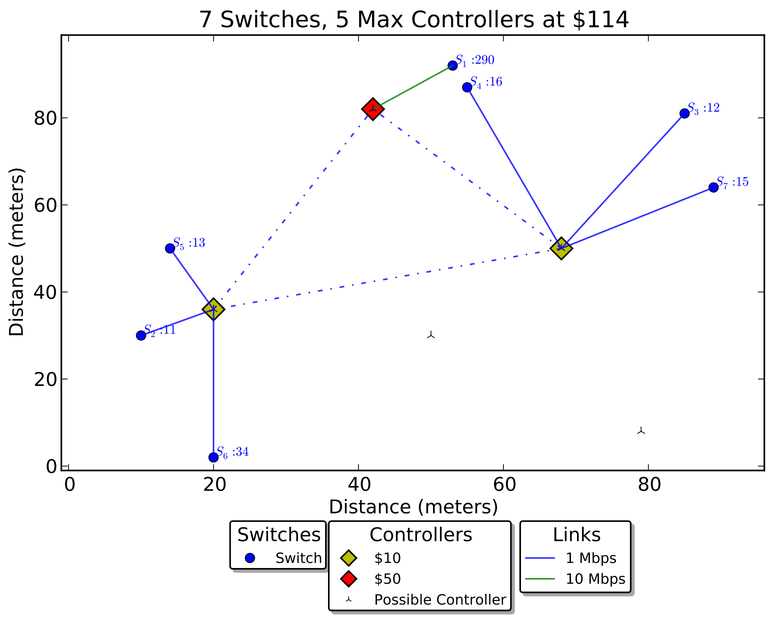


**Figure 4.3:** Optimal solution found by CPLEX for the planning problem example.

We are already aware of the solution to this problem but the optimizer will validate the above solution. Furthermore, the optimizer suggests the time taken to get the solution, which is an important information when the problem size increases. The solution is plotted and shown in Figure 4.3. The links connecting the switches to the controllers are shown with either solid blue or green colour. A blue colour indicates the cost for the links of type one (1 Mbps). The green colour is the cost for the links of type two (10 Mbps). Similarly, the controller placements are coloured and are placed on top of the possible placement markers. The yellow colour indicates the first controller type and the red colour indicates the second controller type. As we can see, the result of the optimization matches the solution found earlier. The links between controllers are always the same type of colour, since we assume that it takes minimal bandwidth to have controllers communicate with each other. Because the goal of the planning model is to place controllers on a network, traffic between controllers is not considered and in our model, the cheapest link speed should be good enough for connecting controllers together.

### 4.1.3 Results for Small to Large Input Sizes

The input to the model is important because the result is a direct reflection of the input. Using the placement problem, we are interested in what happens to our results when the number of switches and the number of possible placement controllers increases. Generally, as the input increases so should the cost to the solutions and the time taken to find the solution. This section will present and analyze the different results given by the planning model.

The optimization should be measured using a variety of methods. One way to measure it is to keep track of the cost for different ranges of solutions. Another method is to keep track of the time taken by the solver to find an optimal solution. The cost of the solution should be increasing as the number of switches increases. This is because the input increase means more controllers may be placed and all those switches must be connected. The number of switches in the topology is directly related to the number of links placed by the solver between switches and controllers. Since every switch must be connected to a controller, it increases the cost of the solution.

The input for the model is very important and in this section, we show what the input for controllers may be. Today many controllers exist and have been tested in real scenarios. Each of these controllers has their own specifications. The specification of each controller type depends on the hardware used and the implementation of the programming language. Table 4.5 shows four types of controllers that are used by the solver.

The specification of the link types that connect controllers and switches together is easy to determine. On any networking sales website, one can determine the cost per meter and the types of links that are available. Currently, a meter of 100 Mbps ethernet is about \$0.25 and \$0.63 for a meter of 1 Gbps ethernet. Furthermore, over 10 Gbps fiber optic cable is on average \$29 per meter. Table 4.6 shows the link input

**Table 4.5:** The type of controllers that are used as input to the planning model.

| | Type 1 | Type 2 | Type 3 | Type 4 |
|---|---|---|---|---|
| **Cost** ($\kappa^c$) | \$1,200 | \$2,500 | \$6,500 | \$12,000 |
| **Ports** ($\alpha^c$) | 8 | 32 | 64 | 128 |
| **Processing** ($\mu^c$) | 2,500 | 4,000 | 8,000 | 15,000 |
| **# Controllers** ($\varphi^c$) | 20 | 15 | 10 | 6 |

**Table 4.6:** The type of links that are used as input to the planning model.

| | Type 1 | Type 2 | Type 3 |
|---|---|---|---|
| **Cost/meter** ($\phi^l$) | \$0.25 | \$0.63 | \$29 |
| **Bandwidth** ($\omega^l$) | 10 Mbps | 1 Gbps | 10 Gbps |

available for the optimization.

The overall input parameters for the optimization are listed in Table 4.7. The area in which the switches and the possible controller locations are generated is 1000 meters by 1000 meters. The reason for the larger area size is because at some point, 200 switches will be generated and a smaller area would not display well. The optimizer is limited to run on a single thread and a maximum of 30 hours of processing time for each scenario. All other parameters are left to their default settings.

Locations of the switches and possible placements are generated randomly for each of the locations within the allowed area. Two values generated makes a single point of $(x, y)$ to be part of the input. The packets ratio to the controllers from each switch ($\sigma^s$) is also generated randomly (see Table 4.7). At random, a set of 10, 20, 30, 40, 50, 75, 100, 150, and 200 switches are generated that should explain what happens to the solution as the number of switches increases. For each of the number of switches in the set, changing the number of controller placement locations shows if it is possible to improve the solution by considering different placement locations. For each set $S$, a new set of maximum placement controllers is generated at random of size 5, 10, 15,

**Table 4.7:** The input parameters for the planning problem.

| | |
|---|---|
| **Maximum Latency** | 250 ms |
| **Packet Size** | 150 Bytes |
| **CPLEX Restrictions** | Single Thread Run |
| **Simulation Area** | 1,000x1,000 meters |
| **Link Available** | Unlimited |
| **Controllers Available** | 51 |
| **Maximum Optimization Duration** | 108,000 Seconds |
| **Switch to Controller** ($\sigma^s$) | Random Integer (100,999) |

and 20 possible controller placement locations. This would mean that 20 switches are optimized using 5 potential locations for the controllers, then with 10 potential controller locations and so on.

Since the number of switches ranges from 10 to 200, when we consider 5 possible controller locations, it is not possible to have feasible solutions to our problems because the processing of controllers is low. For example, if each switch sends 900 packets per second to their future connected controller, it would mean that a network of 200 switches would require 180,000 packets to be traversing the network. The processing power of switches in Table 4.5 for maximum of 5 switches can only handle 75,000 packets per second. This forces us to use a different controller table that has high processing of packets ratio. Table 4.8 shows the controller information that is used when optimizing when $P$ is 5 and $S$ is 10, 20, 30, 40, 50, 75, 100, 150, and 200.

In total there are 36 different problems that must be optimized. Since results can vary between two problems of the same size, we are taking the average over 4 different instances resulting in solving 144 different optimization problems. The smallest problem size contains 10 switches and 5 controller placement locations. The last and largest problem size is when there are 200 switches and 20 controller placement locations.

The aggregated results are shown in Table 4.9 (see Appendix A for all individual results). The first column is the problem number. The second and third columns

**Table 4.8:** The type of controllers that are used as input when there are a maximum of five possible controller locations for the planning model.

| | Type 1 | Type 2 | Type 3 | Type 4 |
|---|---|---|---|---|
| **Cost** ($\kappa^c$) | \$1,200 | \$2,500 | \$6,500 | \$12,000 |
| **Ports** ($\alpha^c$) | 8 | 32 | 64 | 128 |
| **Processing** ($\mu^c$) | 11,500 | 31,000 | 61,000 | 110,000 |
| **# Controllers** ($\varphi^c$) | 20 | 15 | 10 | 6 |

are the number of switches and number of maximum possible placement locations of controllers. Column labeled $|P'|$ is the average number of controllers that have been installed by the solver. Column denoted $|L'|$ is the average number of links installed for the whole network. The average number of controllers installed and the average number of links in the network are rounded up because two controllers and a half cannot be installed. If more than one controller exists, then there are a total of $|L'| = \frac{|P'|(|P'|-1)}{2} + |S|$ links for each scenario. For example, problem 4 has three controllers, meaning that $\frac{3(3-1)}{2}$ links are required to connect controllers using a full mesh topology. Then links to connect the switches to controllers are also required for a total of 13 links as indicated in row four and column $|L'|$. The packets column represents the total number of packets for the network. The last two columns labeled Cost and CPU are the cost of the solution (\$) and the time taken (seconds) by the solver to find the optimal solution. It is important to note that problems 32 and 36 do not represent the optimal solution but rather the best solution found when the time limit (set to 108,000 seconds) was reached. The problems that have at least one instance that exceeds the time limit is shown in brackets. The column "Average" marks which part of the table data are averaged over the four instances.

**Table 4.9:** Results obtained with CPLEX when $|S| = \{10, 20, 30, 40, 50, 75, 100, 150, 200\}$ and $|P| = \{5, 10, 15, 20\}$. The results show the average over 4 instances.

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| | | | Averages | | | | |
| 1 | 10 | 5 | 2 | 11 | 5,060 | 3,227 | <1 |
| 2 | 10 | 10 | 3 | 13 | 5,783 | 4,243 | 1 |
| 3 | 10 | 15 | 3 | 12 | 5,019 | 3,841 | 3 |
| 4 | 10 | 20 | 3 | 13 | 5,169 | 4,086 | 6 |
| 5 | 20 | 5 | 1 | 20 | 11,010 | 4,558 | <1 |
| 6 | 20 | 10 | 5 | 29 | 11,209 | 8,138 | 3 |
| 7 | 20 | 15 | 4 | 26 | 10,181 | 7,512 | 31 |
| 8 | 20 | 20 | 4 | 27 | 10,857 | 7,774 | 83 |
| 9 | 30 | 5 | 1 | 30 | 15,921 | 5,824 | <1 |
| 10 | 30 | 10 | 5 | 41 | 16,391 | 12,747 | 18 |
| 11 | 30 | 15 | 6 | 45 | 17,335 | 13,138 | 155 |
| 12 | 30 | 20 | 6 | 42 | 16,734 | 12,727 | 3,087 |
| 13 | 40 | 5 | 3 | 43 | 21,943 | 8,326 | <1 |
| 14 | 40 | 10 | 6 | 55 | 23,192 | 19,195 | 49 |
| 15 | 40 | 15 | 6 | 55 | 21,044 | 16,980 | 774 |
| 16 | 40 | 20 | 7 | 57 | 23,852 | 19,007 | 5,401 |
| 17 | 50 | 5 | 2 | 51 | 27,096 | 8,781 | <1 |
| 18 | 50 | 10 | 6 | 66 | 27,938 | 23,802 | 66 |
| 19 | 50 | 15 | 7 | 69 | 27,554 | 22,808 | 4,284 |
| 20 | 50 | 20 | 7 | 71 | 27,285 | 22,040 | 1,146 |
| 21 | 75 | 5 | 3 | 78 | 41,773 | 12,992 | <1 |
| 22 | 75 | 10 | 8 | 101 | 42,758 | 38,073 | 292 |
| 23 | 75 | 15 | 8 | 102 | 40,548 | 35,277 | 3,365 |
| 24 | 75 | 20 | (8) | (98) | (40,994) | (35,484) | (58,657) |
| 25 | 100 | 5 | 4 | 106 | 56,447 | 17,638 | <1 |
| 26 | 100 | 10 | 9 | 133 | 55,348 | 49,489 | 498 |
| 27 | 100 | 15 | 11 | 151 | 55,206 | 48,214 | 14,666 |
| 28 | 100 | 20 | 11 | 156 | 54,241 | 47,172 | 18,206 |
| 29 | 150 | 5 | 3 | 153 | 84,221 | 26,803 | <1 |
| 30 | 150 | 10 | 8 | 173 | 80,962 | 73,883 | 2,623 |
| 31 | 150 | 15 | (12) | (211) | (81,141) | (73,102) | (46,219) |
| 32 | 150 | 20 | (12) | (216) | (82,105) | (73,981) | (108,000) |
| 33 | 200 | 5 | 5 | 209 | 110,906 | 33,402 | 1 |
| 34 | 200 | 10 | 10 | 245 | 109,956 | 103,888 | 80 |
| 35 | 200 | 15 | (12) | (262) | (109,908) | (100,128) | (38,475) |
| 36 | 200 | 20 | (11) | (253) | (105,541) | (96,483) | (108,000) |

Problems two to four belong to a set of results that have different potential controller placement locations but have the same number of switches. The time taken for the solver to find the solutions for that set increases in a non-linear form. The first problem takes less than 1 second and the last run for that set of switches takes 6 seconds. In general, as the input size ($|S|$ and $|P|$) increases, the time taken to find the optimal solution also increases because the model must consider more scenarios before deciding where to place controllers and how to connect the controllers and switches together. However, this is not always the case. For example, problem 20 takes less time to solve than problem 19 even though it has more potential locations for the controllers. A reason why the optimizer may take less time for larger input size rather than smaller input is because the optimizer uses the branch and bound algorithm to try different combinations to find the optimal solution and heuristics are used to shorten the combinational space by finding the solution to the relaxation of the current node [45]. The time taken to find an optimal solution for problems 32 and 36 was interrupted for all instances because the maximum running time was reached. This means that the cost for those problems may not be the optimal solution as better solutions may have been found if more running time had been available.

The cost of the solutions starts with a high value because the controllers' starting price is $1,200. Since the first problem has placed two controllers, and if the controllers are of type one, the total cost for only the controllers is $2,400 - in addition, about $106 is the cost of connecting controllers and switches together. The results show that the first four problems have costs between $3,227 and $4,243. These costs are close to each other because each of the problems 2 to 4 have three controllers installed and it also shows that switches may be placed far apart for the more expensive scenario. Similarly, problems 5 to 8 have costs between $4,558 to $8,138. The reason why the optimal costs are not close to each other is because the input is generated randomly and different distances between switches and controllers may exist. Also, the result for problem 5 uses a different controller input table that can process more packets than the input for problems 6 to 8. These differences are small compared to the differences noted in problems 29 and 32. The cheapest optimal solution is $26,803 and the most expensive is $73,981.

Until now, we have been comparing the cost of the optimal solutions for different problem sizes but this does not explain the reliability of the results. Exploring the reliability of the data can be done in various ways but one way is to find the confidence interval. For our results, a confidence interval is stated as a percentage with a value of 95%. The percentage represents the likelihood that another sample will fall into the same interval. In Figure 4.4, the vertical segments along the graph show the confidence interval. The points in the plot indicate the mean cost for each set of switches grouped by the number of maximum possible controller locations ($|P|$). As expected, the cost of solutions increases as the input size increases as shown in the figure. However, the line when $|P|$ is five shows that the maximum cost is about \$33,502 and this is why the line ends half way in the graph. Generally, the line when $|P|$ is five should be together with the other lines but the result show that the line is far apart from our other results. This is much expected because the controller input table for when $|P|$ is five can process a lot more packets than for other $|P|$ instances. It means that less controller have to be placed in the network and resulting into a cheaper overall price. The largest confidence interval is on problem 35 with $\pm$\$8789. The results indicate that confidence bars are very narrow for this graph and it shows that the optimal placement cost of new optimizations may lie within narrow intervals. Therefore, the optimization cost reported by CPLEX is considered reliable.

When the error bars are wide, it indicates that the results are less reliable because the range covers a wider set of values. This is shown in Figure 4.5. We can improve the error bars by running more than four instances for each problem. Decreasing the error bars is possible if we decrease the standard deviation since standard deviation is related to the square root of the number of instances each problem is repeated. Looking at the figure, when $|P|$ is 20 and $|S|$ is 100, the error bars are very wide. To make the bars shorter, we need to make the standard deviation smaller by running each problem more than 4 times. If we wish to decrease the standard deviation by half, each problem would have to run 16 times. When $|P|$ is 20 and $|S|$ is 100, each problem instance takes 16 hours to complete. This would mean that for 16 runs, it would take 256 hours to complete all the 16 instances. The standard deviation for four instances is 8,490 seconds and the confidence interval is $\pm 27,019$ seconds. Assuming the same mean is used, if we were to run the problem 16 times, the confidence interval would decrease to $\pm 9,056$ seconds (because standard deviation decreased by half). This is 33% of the confidence interval when we compare it to the four instances from our result.

Also, in Figure 4.5, we can see that the error bars are wide for $|P| \geq 15$ and

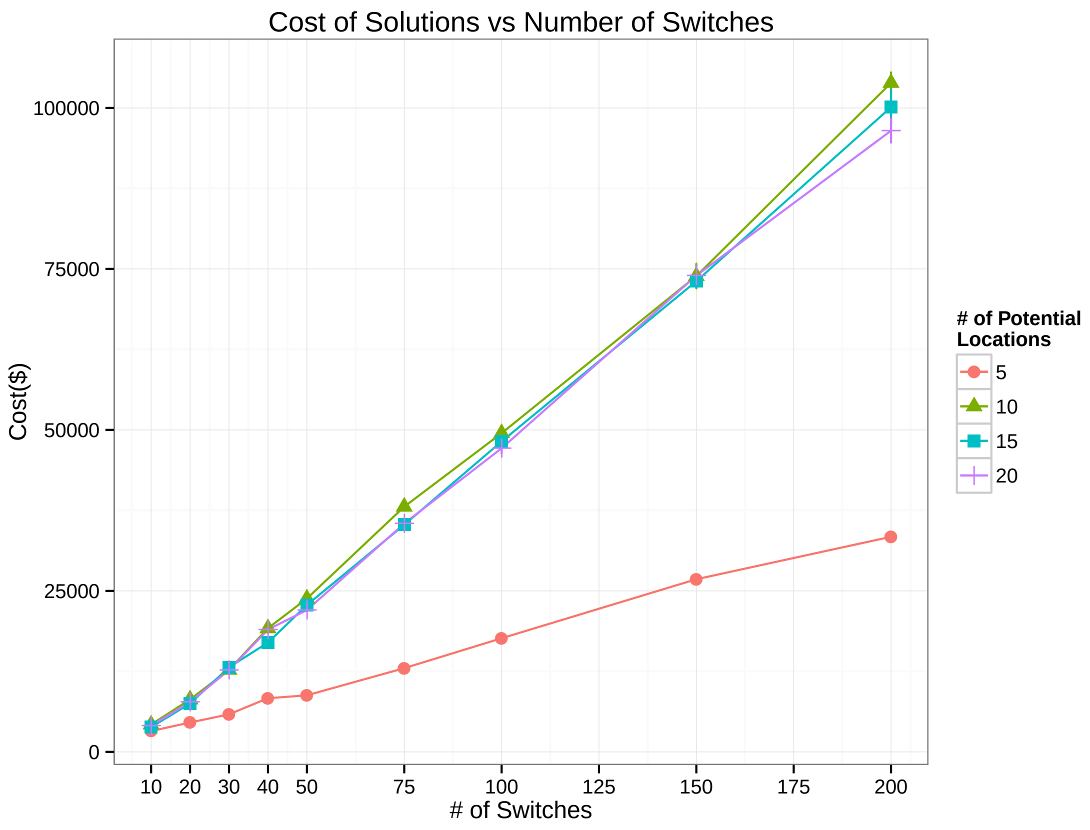


**Figure 4.4:** Cost of solutions colored against maximum number of controllers with 95% confidence interval.

$|S| \geq 50$. We can improve the reliability of the result for problem 31 by running an instance of the problem 16 times. Since the average time for that problem is about 13 hours (46,219 seconds), running it 16 times may take 205 hours (15 fold increase). The results show that the total time taken to find optimal solutions for every problem run four times is 19 days (1,656,776 seconds). Improving the reliability by running all of the problems 16 times is not practical because it would have taken 306 days to find the solutions. The solution time was expected to be high because integer programming problems fall into NP-Hard problem types [46].

Figures 4.6 to 4.9 show the plots of the results grouped by $|P|$. The plots show the solution time and solution cost against the number of switches. The figures show more details than the figures that have all the $|P|$ in one graph. The reason for this is that when $|P|$ is small, the range of values for the solution time and the cost are much lower than when $|P|$ is of a higher value and this allows us to better analyze the

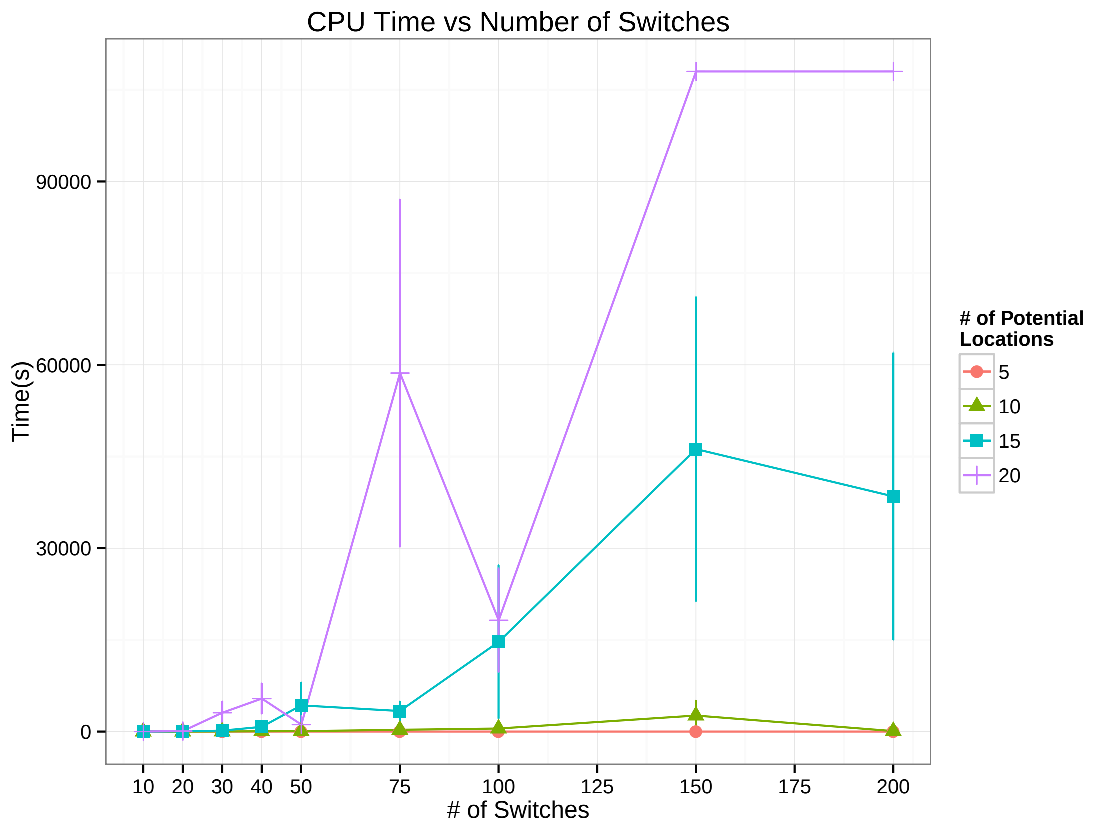


**Figure 4.5:** Solutions time against the number of switches with 95% confidence interval.

graphs. For example, if we look at the time taken to find an optimal solution when $|S|$ is 30 and $|P|$ is 5, we can not see any details in Figure 4.5 but we see considerable details in Figure 4.6.

When there are a maximum of 5 controllers to be placed for all different sets of $|S|$, we have used a different controller input table (see Table 4.8). This means that the packet processing is much higher for this result set. In Figure 4.6, the reason the time spikes when $|S|$ is 50 and $|P|$ is 5 is because for all three problem instances, the solver takes about 0.32 seconds to find the optimal solution. However, when $S$ is between 60 and 100, the optimal solutions are found in a shorter time. When $S$ is 200, the time taken is about 0.67 seconds. Figure 4.7 shows the time and cost of the

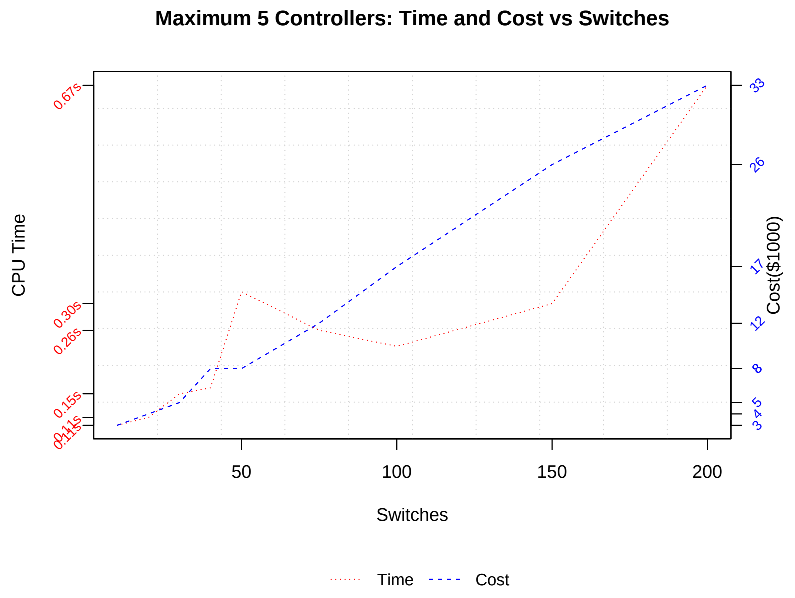


**Figure 4.6:** Solution time and cost vs number of switches for a maximum of 5 controller placement locations.

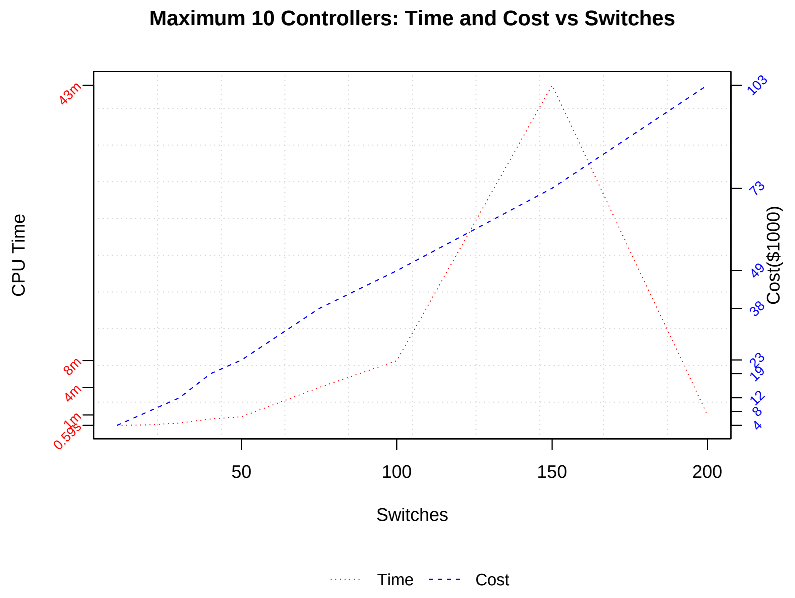


**Figure 4.7:** Solution time and cost vs number of switches for a maximum of 10 controller placement locations.

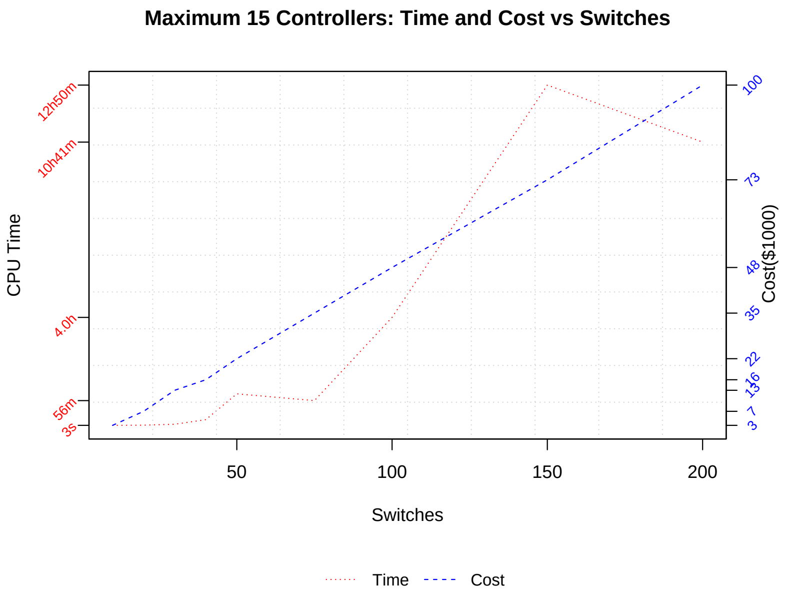


**Figure 4.8:** Solution time and cost vs number of switches for a maximum of 15 controller placement locations.

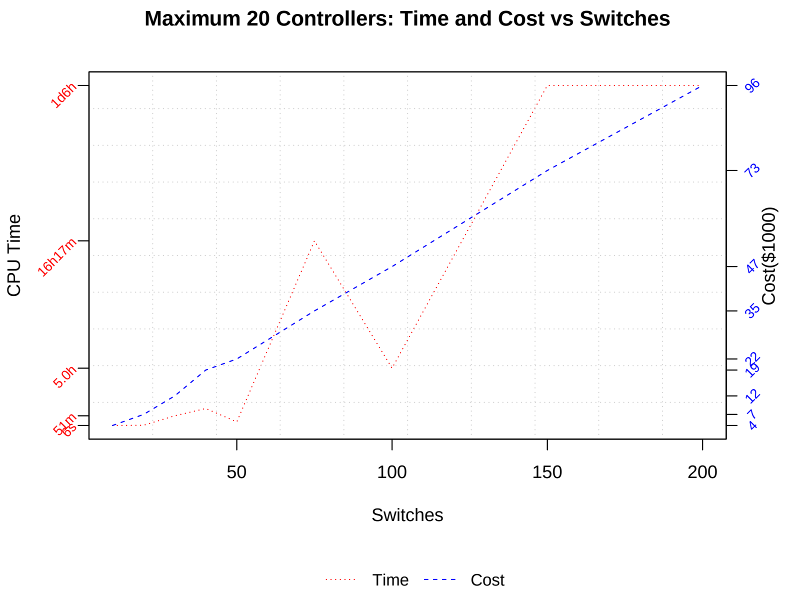


**Figure 4.9:** Solution time and cost vs number of switches for a maximum of 20 controller placement locations.

optimal solutions of the average of four instances when $|P|$ is 10. We can see that the cost increases in a linear form but the time of the optimal solution is different. When $|S|$ is 150, the time taken to find the optimal solution is the longest at about 43 minutes. This is due to the fact that the four problem instances take 2 hours and 48 minutes to find the optimal solution. Interestingly, when $|S|$ is 200, the time taken to find the optimal solution is 79 seconds. The four instances averaged to 79 seconds for that problem.

The solution time in Figure 4.6 to Figure 4.9 shows an increasing pattern in non linear form. It shows that there is randomness when time is reported between the different sample sizes. This relates to the theory that the method of eliminating combinational computations when finding optimal solutions is done using a heuristic method and different times are reported for the same problem sizes. However, if we compare the cost of the optimal solutions between all the figures, we can see a pattern appearing for our data set. All of the graphs show that cost is increasing in linear form. When we observe Figure 4.8, we see that the cost increases similar to a linear model. The optimal solutions are found within seconds and then jump to an hour when $|S|$ is 50. When $|S|$ is past 75, the time increases 13 fold. The longest optimization time is when $|S|$ is 150 with a 12 hours and 50 minutes duration. The reason why such a high optimization time exists for $|S|$ at 150 is because on the third instance of the experiment, the optimization time passes over the 30 hours limit. The other optimizations are much lower. This is shown in Figure 4.5 with wide error bars which indicates that with a 95% confidence, future repeats of the experiment (with different data sets) may lie within the interval indicated by the bars. The result is not reliable because it indicates that any future optimization will fall within a wide set of possible results. When $|S|$ is 200, the optimizer takes two hours less compared to when $|S|$ is 150. Similarly, for the fourth run when $|S|$ is 150; the optimizer passes the allowed time limit. This single instance is over time limit, which raises the average running time from 7 hours to 10 hours.

The massive difference in solution time is explained by the wide error bars in Figure 4.5. The time and cost when $|P|$ is 20 is shown in Figure 4.9. For this problem set, the time changes considerably for every value of $|S|$. When $|S|$ is 30, the highest optimization time is 2 hours and 18 minutes and the lowest is 6 minutes. When $|S|$ is 40, the highest solution time is over 3 hours and the lowest is 15 minutes. When $|S|$ is 50, the optimal solution is found in 15 minutes. The different ranges of results for the four instances when $|S|$ is 50 are much closer to each other compared to the previous $|S|$ sets. This is verified by Figure 4.9 where the error bars are much wider when $|S|$ is 30 and 40 compared to when $|S|$ is 50. The difference of the solution times for four instances (when $|S|$ is 75 and $|P|$ is 20) are: about 1 hour and 45 minutes for the shortest time and the longest time reaches the limit of 30 hours. Instance one and three both reach the maximum time. When $|S|$ is between 150 and 200 and $|P|$ is 20, for all the problems and all instances, the maximum time is reached. Since all the problem instances are over the time limit, this means that there would be no error bars visible because there is no difference in solution time.

### 4.1.4 Results for Increasing Possible Controller Locations

In the previous section we investigated how the time and the cost of the solutions were affected by increasing the input size. Both the set of the switches and the set of possible controller placements were increasing. In this section, we will explore what happens to the cost and time of optimal solutions when the number of switches remains the same and the number of possible controller locations increases. The goal is to observe a pattern that flat lines for the optimal time and cost to find the solutions. A set of 20 random switches is generated for every problem because in the previous section we saw that for all instances when $|S|$ is 20 an optimal solution was found.

The controller and link information remain the same as in the previous section (see Table 4.5 and Table 4.6). Furthermore, the overall input parameters to the model remains the same as well (see Table 4.7). Since there are only 20 switches generated for the topology, the packets that leave from each switch will also have to be generated ($\sigma^s$). A random integer between 100 and 999 is assigned for every $\sigma^s$. The different problems are optimized when $|S|$ is 20 and $|P|$ is 5, 10, 15, 20, 25, 30, 40, 50 and 75. In total, there are 9 problems that will be repeated 4 times.

The procedure to find optimal solutions for the problems defined above is the same as the one described in the methodology. The optimizer returns the optimal cost and time for each of the problems and is shown in Table 4.10 (see Appendix B individual results).

**Table 4.10:** Planning results obtained by CPLEX for 20 switches and $P = \{5, 10, 15, 20, 25, 30, 40, 50, 75\}$.

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| | | | Averages | | | | |
| 1 | 20 | 5 | 5 | 28 | 11,822 | 12,071 | < 1 |
| 2 | 20 | 10 | 5 | 28 | 11,301 | 10,556 | 3 |
| 3 | 20 | 15 | 4 | 26 | 9,809 | 9,498 | 140 |
| 4 | 20 | 20 | 5 | 28 | 11,124 | 10,852 | 21,563 |
| 5 | 20 | 25 | 4 | 27 | 11,335 | 11,073 | 6,793 |
| 6 | 20 | 30 | 4 | 26 | 10,576 | 10,001 | 1,356 |
| 7 | 20 | 40 | (5) | (27) | (11,396) | (10,423) | (27,616) |
| 8 | 20 | 50 | (4) | (27) | (10,539) | (10,294) | (54,637) |
| 9 | 20 | 75 | (4) | (27) | (10,716) | (10,416) | (82,405) |

The result table shows the number of switches ($|S|$) and the number of maximum possible controller placements ($|P|$). The output of the solutions includes the average number of placement locations where controllers are installed ($|P'|$), the average number of links that were picked by the optimizer ($|L'|$) and the average number of packets that must traverse in the whole network from the switches to the controllers. Finally, the minimum cost and the time taken by the optimizer to find the optimal solution are shown in the last two columns. It is important to note that $|P'|$ and $|L'|$ are rounded up because we cannot have a quarter of a controller or a quarter of a link installed.

We notice that the first problem has the most expensive solution at $10,000, whereas the cheapest solution is $9,498 (problem 3). If we examine the individual instances for the first problem, we see that the fourth instance is over $14,000 and the next two cheapest instances are over $12,000. By examining individual instances for problem 3, we can see that it has a minimum cost of about $7,000 and a maximum cost is $10,000. However, the rest of the problems are within $10,500. The results indicate that the cheapest solution is found by problem number 3. The cost of the optimal solution tends to become stable as the number of controller locations increases, this is true especially by the end of our results. The time taken by the optimizer to find the optimal solution for problem number one is within a second. As the number of potential locations increases, the time tends to be increasing because the solution to problems 1, 2, 3 and 4 are in increasing order. However, problem 5 and 6 interrupt this pattern. Then the pattern continues where the time taken to find the optimal solution increases for problems 7, 8 and 9. The same problems optimized exceed the time limit of 30 hours (108,000 seconds) with a least one of the optimization instances. Therefore, the results numbers in parenthesis shows which problems experience the solver going over time limit.

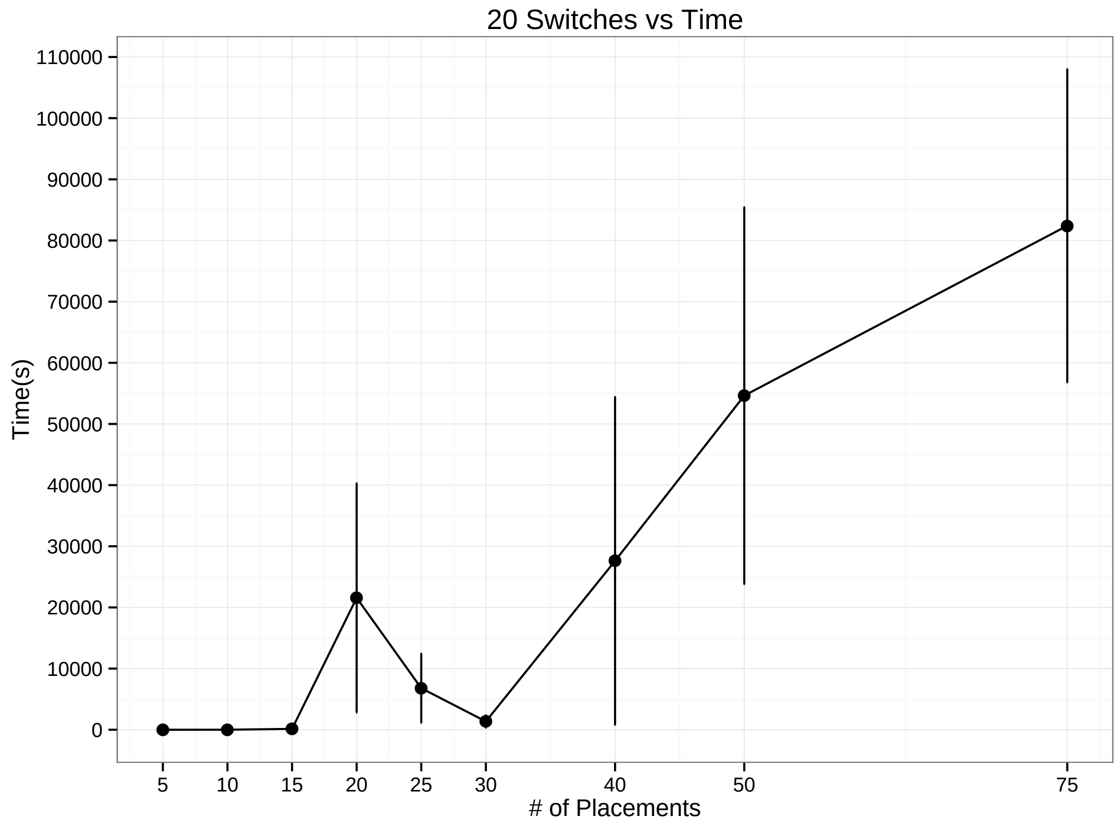


**Figure 4.10:** Solution time against the same number of switches and increasing potential locations with 95% confidence interval.

With a 95% confidence interval, Figure 4.10 shows the plot of the average optimal time taken to find the solutions. For the first three potential placements locations, the time taken is between 0.11 seconds and 140 seconds and the reason why lines are straight and without error bars is because the averages and confidence intervals are small compared to the graph interval for the time axis. The time axis' interval is every 2 hours and 46 minutes (10,000 seconds). The next interval is when $|P|$ is 20 and that takes about 5 hours and 33 minutes and here we see a wide error bar. With 95% confidence, if new problems are optimized, their optimal solution time could be within 3,000 and 40,000 seconds. This is equivalent to 1 hour and 11 hours. This is a wide range indicating that it is a weak estimate of the true population mean (when $|S|$ and $|P|$ is 20). The next plot points show a decreasing optimization time to find the optimal solution. Also, they show shorter error bars. When $|P|$ is 30, the error bars are not visible but they exist. This is because the range is very small and it is not visible on the graph, the confidence interval is $\pm 3,231$ seconds from the mean. When $|P|$ is 40, the confidence interval is $\pm 85,277$ seconds (23 hours). The reason for such a large interval is because one of the four instances went over the time limit of 30 hours. There are instances for when $|P|$ is 50 and 75 where time is exceeded by the optimizer; therefore, the average time increases as a result.

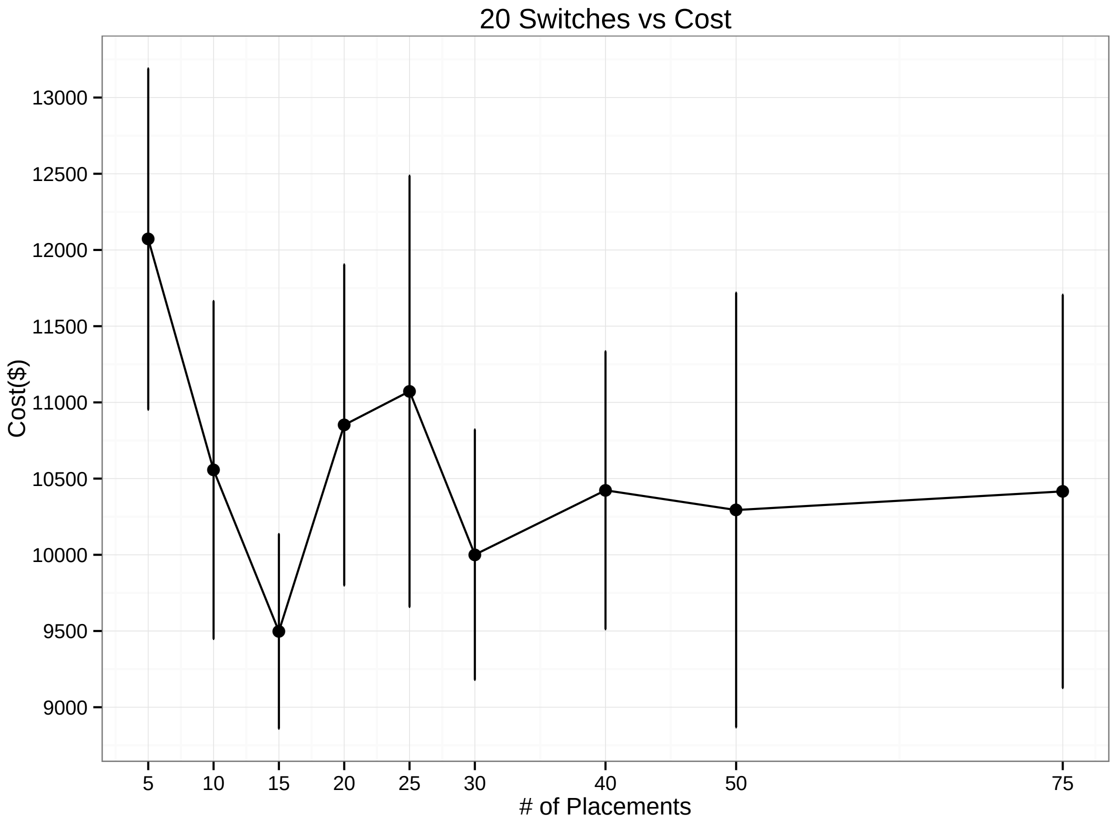


**Figure 4.11:** Cost of the solutions while increasing the potential locations and keeping the number of the switchs the same with 95% confidence interval.

From the results table shown above, the cost of the solutions are around \$10,000. We need to know how likely it is that the mean cost for each $|P|$ represents the overall optimization costs. Figure 4.11 shows the plot of the cost against different possible placement location sets. With 95% confidence level, the graph displays the range of values for each $|P|$ that the next optimal cost will fall between. All of the error bars are wide and the confidence interval with highest range is ±4,536 (compared to ±8,789 from the previous section). Generally, as the confidence interval gets wider, the accuracy of the sample mean as an estimate of the population mean gets worse. In this case, this is likely due to the fact that there are only 4 samples to draw the confidence intervals from. To improve the accuracy of the estimate, we must increase the number of times each problem is optimized and to do so, as we have seen in the previous section, we must run our optimizations 16 times. If we repeat the same experiment 16 times instead of 4 times, the optimization would finish within 40 days. From the graph, we can see that when $|P|$ is 15, the lowest cost (\$9,498) is found. Since the optimizer run time has passed the allowed time limit for two of the four instances when $|P|$ is 50 and 75, we can discard the result because it does not provide accurate information since as the optimal cost was not always found.

In conclusion, we can see that the optimal cost (when reported) is always more accurate than the optimal time of the solutions. As the number of switches and potential controller locations increases so does the cost of the optimal solution. Also, the cost of optimal solutions tends to improve if the number of potential locations is increased. In any case, as the input size increases, the time to find an optimal solution is also increased.

## 4.2 Expansion Problem: Results and Analysis

In this section, the expansion model is first explained with the same example as in Section 4.1.2. The difference is that changes to the existing network are made. Expansion of a network may involve adding or removing switches, changing the controllers by adding, removing or completely replacing controllers or just improving the links between controller and switches. Since we take into consideration the flow setup latency, one change could be to lower the latency to ensure flow setup times decrease for a better experience. Also, a network may change its policy to have deep packet inspections with the full packet size being sent to the controllers and the expansion model takes this into consideration.

### 4.2.1 Methodology

The expansion model has the same process as the planning model and expects the same input format but there are some differences between the two. When we consider an expansion scenario, if we compare the processes of the planning model (see Figure 4.1) to the expansion model, the input is changed and few additional functions are added to be able to find the optimal solution.

If the expansion model is used to make changes to an existing network, then we would need the results of the planning model to be included before the expansion model can be generated. This is due to the fact that current installation of the switches and links would have to be included into the linear program. Thus, making us have additional implementation functionality when generating the model. In addition, input to generate the model may be different because the controller and possible controller locations may be different. Figure 4.12 shows the expansion model process. These additional steps are shown with the white boxes in the figure. Since our expansion model is a generalization of the planning model, if we were to place controller(s) for the first time using the expansion model, the process would be the same as in the planning model because problem input is the initial input as in the planning model. Furthermore, there are no planning results which are added to the expansion model.

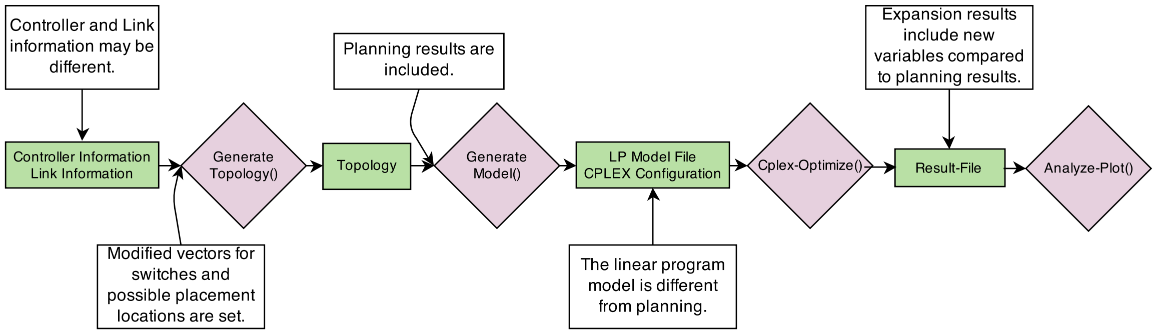

Controller and Link information may be different.
Planning results are included.
Expansion results include new variables compared to planning results.
Controller Information Link Information
Generate Topology()
Topology
Generate Model()
LP Model File CPLEX Configuration
Cplex-Optimize()
Result-File
Analyze-Plot()
Modified vectors for switches and possible placement locations are set.
The linear program model is different from planning.

**Figure 4.12:** Steps to find the optimal solution for expanding an existing SDN network.

Generating the model and plotting functionality is the major difference. If an expansion model is chosen, then the model will include the mathematical model from the expansion. This in turn has a new objective function and additional constraints. In the case of plotting an expansion solution, additional switches are marked on the graph and including the links that are placed by the expansion model in Section 3.3.

### 4.2.2 Restrictions on Input for Expansion

Our implementation does not cover incorrect changes to the input format. If such changes are made, the result would not be accurate. The vector that holds the switch location information must remain the same for the expansion model for the most part. Furthermore, certain scenarios are allowed when changing any of the existing vectors. The location $(x, y)$ of the switches in a network may change and its outgoing packet rate to the connected controller may also change. If switches are to be removed, the only possible way to remove them is by shortening the vector at the end. Removing elements from the vector and adding new elements will not work. Any other location is not possible as it will supply incorrect installation information from current installation of the switches and controllers to the expansion model. If any changes are made before the expansion model is generated, the index location of each switch would not match the switch vector of the planning model. For example, if the vector $S$ had values $\{(1,2),(3,4),(8,7)\}$ in the planning model and in the expansion model switch $(1,2)$ is removed and $(5,9)$ is added, then the vector for expansion would become $\{(3,4),(8,7),(5,9)\}$. The installation information from the planning model for switch $(3,4)$ is now moved to $(8,7)$ which is incorrect. To handle these scenarios, we must have implementation support in software by analyzing previous and current vectors and regenerating placement information for expansion. However, this is beyond the scope of this research and does not affect the generality of the model.

**Table 4.11:** Allowed changes to the controller input for expansion.

| | Type 1 | Type 2 | Type 3 | Type 4 |
|---|---|---|---|---|
| **Cost** ($\kappa^c$) | \$10 | \$50 | \$80 | \$100 |
| **Ports** ($\alpha^c$) | 4 | 8 | 16 | 32 |
| **Processing** ($\mu^c$) | 100 | 300 | 400 | 500 |
| **# Controllers** ($\varphi^c$) | 0 | 0 | 5 | 8 |
| | $\longleftrightarrow$ Not Used | | $\longleftrightarrow$ New Controllers | |

We assume that the possible controller placement locations vector must not change much. The only allowed changes in this vector is to add at the end of the vector. Furthermore, the vectors that hold information about the controllers and link types cannot be removed or reordered. The only possible way to remove inventory in the case of controllers is by setting the number of controllers available to zero. Table 4.11 shows the allowed changes to the controllers. New controllers added are shown in the column marked "New Controllers". Removing links is a little more complicated because there is no limit to how much link can be used for each link type. To make the model not choose a link, its link bandwidth must be zero ($\omega^l$). This should force the optimizer to pick other links because the bandwidth of the link must be greater than the bandwidth that is required by the switch. Table 4.12 shows the correct changes to the links table. The first type is disabled, the second type has its bandwidth and cost changed and finally a new link of type 3 is added.

**Table 4.12:** Allowed changes to link input for expansion.

| | Type 1 | Type 2 | Type 3 |
|---|---|---|---|
| **Cost/meter** ($\phi^l$) | \$0.05 | \$6.5 | \$12.31 |
| **Bandwidth** ($\omega^l$) | 0 Mbps | 1.8 Mbps | 1 Gbps |
| | $\longleftrightarrow$ Not Used | $\longleftrightarrow$ Changed | $\longleftrightarrow$ New Link |

### 4.2.3 Cost of Removing a Link or Controller

One of the main factors that an expansion model uses to decide the optimal placement of controllers is the cost of removing links and controllers. The cost of removing a controller will normally include the time taken by a technician to reconfigure the network, and may involve the physical reallocation of the device as well as efforts to ensure that the controller is stable. This value is specified in a dollar amount because it will be one of the costs that will be added to the cost function.

The value to remove an item from the existing network could take many forms. In our implementation when we generate the linear problem for CPLEX, an expansion cost multiplier is used, which is a variable that indicates how much more it would cost to reallocate a controller or remove a link compared to the installation price for placement. This is a value that can range from 0.01 to 1000 and that is multiplied to the actual cost to install the controller or link.

**Table 4.13:** The effect of the planning cost of $13.00 under different cost-multiplier scenarios.

| | Planning($) | Cost-Multiplier | Expansion($) | Network Changes |
|---|---|---|---|---|
| Example 1 | 13.00 | 1000 | 13000 | Keeps the old structure |
| Example 2 | 13.00 | 100 | 1300 | Keeps the old structure |
| Example 3 | 13.00 | 10 | 130 | Few changes |
| Example 4 | 13.00 | 1 | 13 | More changes |
| Example 5 | 13.00 | 0.1 | 1.30 | More changes |

Optimal expansion is highly influenced by the cost multiplier variable. Table 4.13 shows a sample of the cost and what it might mean to the optimizer when deciding to remove a link and place it again. If the desire is to keep much of the current network structure and avoid changes, a higher cost should keep the structure of the current network. The last column indicates what may happen if a specific cost multiplier is used.

### 4.2.4 Detailed Example

In this section, we will be going over an expansion example of an existing SDN network. The example from the placement section will be used to show how the generalized model of expanding a network would go through both original and new constraints. From the example in Section 4.1.2 we will be adding a new switch and changing $\sigma^4$ to one packet per second. The new switch will be located at (80,14) and will send 16 packets per second to the controller ($\sigma^8$). Figure 4.13 shows the new changes before the expansion is performed. In the expansion figure, we can see that on the legend, a new label that marks new switches was added to the network.

The same procedure as the placement example will be followed for the expansion example. First, the example will go over the same constraints that we had in the placement example and then the new constraints that decide if an existing resource is removed or reallocated.

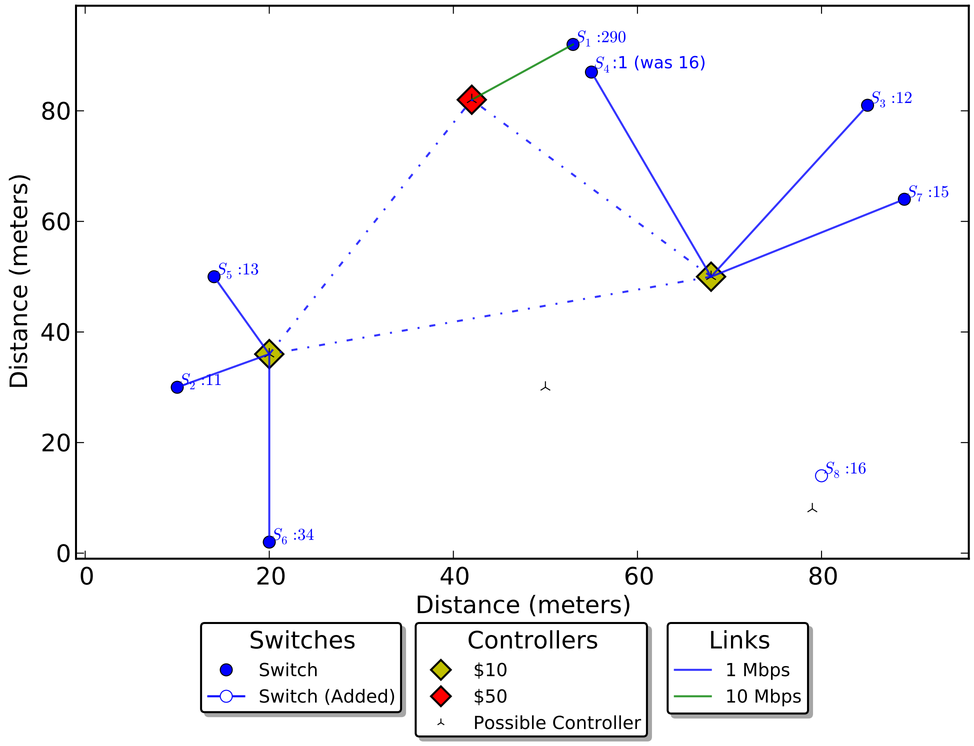


**Figure 4.13:** The modifed SDN toplogy before the expansion example. Changes are made by adding $S_8$ and changing $\sigma^4$ to 1.

### Controller Processing Capacity

Equation 3.7 still validates the incoming requests to the possible controller placements. The constraint still runs exactly the same as in the example explained above and the switches at locations 2, 5 and 6 are still allocated to $P_4$ because nothing from those switches has changed. However, as we have seen previously, $S_1$ was assigned to $P_2$ but $S_4$ was not. Switch $S_4$ sends one packet per second to the controller, the controller on location $P_2$ can now have $S_4$ connected to it. This is the only difference. The switches 3, 7 and 8 can also be linked with $P_5$ because it is the closest and a controller of type one can process all the requests from the switches. The optimizer already produces the same result when considering $S_3$ and $S_7$, in addition it is also suggested that $S_8$ be added to $P_5$.

The total processing of packets that will have to be done at $P_2$ is 291 packets per second and at $P_5$ 43 packets per second. Finally, $P_4$ stays at 58 packets per second. The controller in location $P_2$ has packets coming in faster than what the controller of type one can handle. Therefore, $P_2$ installs a type two of the controller, whereas the other placement locations install type one of the controller because they require below 100 packets per second. The installed controller types remain the same as in the placement example.

### Controller Port Availability

The port availability is verified by the constraint in Equation 3.5. The constraint concludes the same as in the placement example. There are three controllers that must be connected using a full mesh topology and they will be using $|P'| - 1$ ports. Since 3 controllers are installed, there will be $3 - 1$ ports used by each controller to create a full mesh connection. Additionally, each controller must be linked to its assigned switches. Installed controller at location $P_2$ has three switches connected which results in 5 ports used. The same applies for $P_5$. However, $P_2$ only has two switches connected and this means that the switch will be utilizing 4 ports only. This constraint passes because all the controllers that are installed do not exceed the number of ports available for use.

### Controller Inventory Limits

Equation 3.8 limits how many controller types are installed. Throughout the optimization, the solver ensures a controller is installed at optimal locations. By the end of the optimization, the solver also checks for each installed controller type if it has not exceeded the number of available controllers listed in Table 4.1.

### Link Bandwidth Between Switch and Controller

The link bandwidth is verified the same way as in the placement example. An extra check is performed for the new switch ($S_8$) and since this switch only sends 16 packets per second, it is sufficient to use a link of type one.

### Controller Connectivity

Equation 3.11 ensures that the controllers are connected together. The same connectivity is maintained throughout the expansion phase because the location of controller placements did not change and the number of controllers installed did not change. Therefore, the expansion scenario produces the same result.

### Logical Constraints for Expansion

One reason we can use the expansion model for placing a new set of switches is because of the logical constraints in the mathematical model. If a network already exists, we use the information from the planning to provide information to the model that will help when doing the logical constraints.

Comparing the placement phase and the expansion phase of controller connectivity, we notice that there is a reallocation of $S_4$ from placement $P_5$ to the new placement $P_2$ ($v_{42}^1$). In addition, the new switch $S_8$ is added to controller at location $P_5$ ($v_{85}^1$). Equation 3.23 is used to decide what happens to the existing links if $\bar{v}_{45}^1$ is set. The equation indicates that a link can only be removed if it is set. In our case, link between switch 4 and possible controller location 5 is set because in the planning phase, this link was placed to form an optimal solution. Since the switch is now connected to an installed controller at location 2, the existing link to controller location 5 must be removed. The equation below shows how an existing link is removed. We know that a link between switch 4 and controller location 5 can not be kept anymore and the variable $v_{45}^1$ is forced to 0. The only possible method to satisfy the constraint is to set $Rv_{45}^1$ to 1. The cost to remove an existing link is added to the objective function in equation 3.18.

$$Rv_{45}^1 + v_{45}^1 \geq \bar{v}_{45}^1$$

$$1 + 0 \geq 1$$

Below, we also show the constraint that checks if a link can be kept if it is installed.

Again, since we know that the existing installed link $v_{45}^1$ must be set to 0, the only possible way to satisfy the constraint is to set both sides of the variables to 0.

$$v_{45}^1 - Kv_{45}^1 \leq \bar{v}_{45}^1$$

$$0 - 0 \leq 1$$

We know that for the new connection between switch 4 and the installed controller at location 2 is set ($v_{42}^1$). The same constraint that checks if a link can be removed is considered again (Equation 3.23). This time, since an existing link is not present ($\bar{v}_{42}^1$), the constraint is satisfied only when $v_{42}^1$ is set. This implies that a non existing link can not be removed and $Rv_{42}^1$ is not set.

$$Rv_{42}^1 + v_{42}^1 \geq \bar{v}_{42}^1$$

$$0 + 1 \geq 0$$

Currently connection between switch 4 and its controller does not exist. A link between switch 4 and controller 2 must be placed. The equation bellow shows how a new link is included in the expansion network. Since this is a new link, the installed value of this link from the planning phase is 0 ($\bar{v}_{42}^1$). The inequality bellow forces the left hand side to be set to 1 so that it can be satisfied. This implies that $Kv_{42}^1$ is also set. The expansion cost calculation in Equation 3.18 includes $Kv_{sp}^l$ to the cost calculation.

$$v_{42}^1 - Kv_{42}^1 \leq \bar{v}_{42}^1$$

$$1 - 1 \leq 0$$

Above we considered scenarios when an existing link of a switch to a controller is removed and then added elsewhere. In this scenario, we consider what happens when we add a new switch to the network. We know the optimal location to connect switch 8 is at installed controller location 5 ($v_{85}^l$). Also, we know that this is a new switch and it could have not been installed in the planning phase; therefore, $\bar{v}_{85}^l$ is set to 0. The constraint to remove the existing link is ignored because just by having $v_{sp}^l$ set, the inequality is satisfied.

$$Rv_{85}^1 + v_{85}^1 \geq \bar{v}_{85}^1$$

$$0 + 1 \geq 0$$

Connecting switch number 8 to a controller is performed with the inequality bellow. Since $\bar{v}^1_{85}$ is set to 0, both $v^1_{85}$ and $Kv^1_{85}$ is set to 1 to satisfy the inequality.

$$v^1_{85} - Kv^1_{85} \leq \bar{v}^1_{85}$$

$$1 - 1 \leq 0$$

The list bellow summarizes the variables that are picked by the solver when running the expansion example. The underlined variables

- Controller Placement ($x_{cp}$): $x_{22}, x_{15}, x_{14}$
- Switch and Controller Connectivity($v^l_{sp}$): $v^2_{12}$, $v^1_{75}$, $v^1_{54}$, $v^1_{64}$, $v^1_{24}$, $v^1_{35}$, $v^1_{42}$, $v^1_{85}$, $Kv^1_{85}$, $Kv^1_{42}$, $Rv^1_{45}$.
- Full Mesh Controller Topology($z^l_{pq}$): $z^1_{42}$, $z^1_{52}$, $z^1_{54}$

**Cost of the solution**

From the expansion model and Equation 3.3.2, we can see that only the $K$ and $R$ variables make up the cost. Since we are expanding an existing solution, the only changes suggested (variables that are set and are included into the cost function) are: $Kv^1_{85}$, $Kv^1_{42}$, and $Rv^1_{45}$. From the solver, we know the costs of the variables are formed as follows:

- \$0.696 for $Kv^1_{42}$ ($Kv^l_{sp}$)
- \$1.897 for $Kv^1_{74}$ ($Kv^l_{sp}$)
- \$196.087 for $Rv^1_{45}$ ($Rv^l_{sp}$)
- **Total:** \$198.68

The total cost of \$198.68 for the expansion example is the cost for removing a link and adding two links. The most expensive cost for the solution of the expansion example is \$196.08 when removing a link. This is because the cost of removing a link was a multiplied by 100. A cost multiplier of 100 times the placement cost keeps the network structure intact as much as possible. Finally, the optimal network, once the expansion is completed, is shown in Figure 4.14. The legend indicates which links where affected by the expansion model. In this case, links to $S_4$ and $S_8$ were affected by the expansion. Since we already know from the example, $S_4$ has been moved but $S_8$ is new.

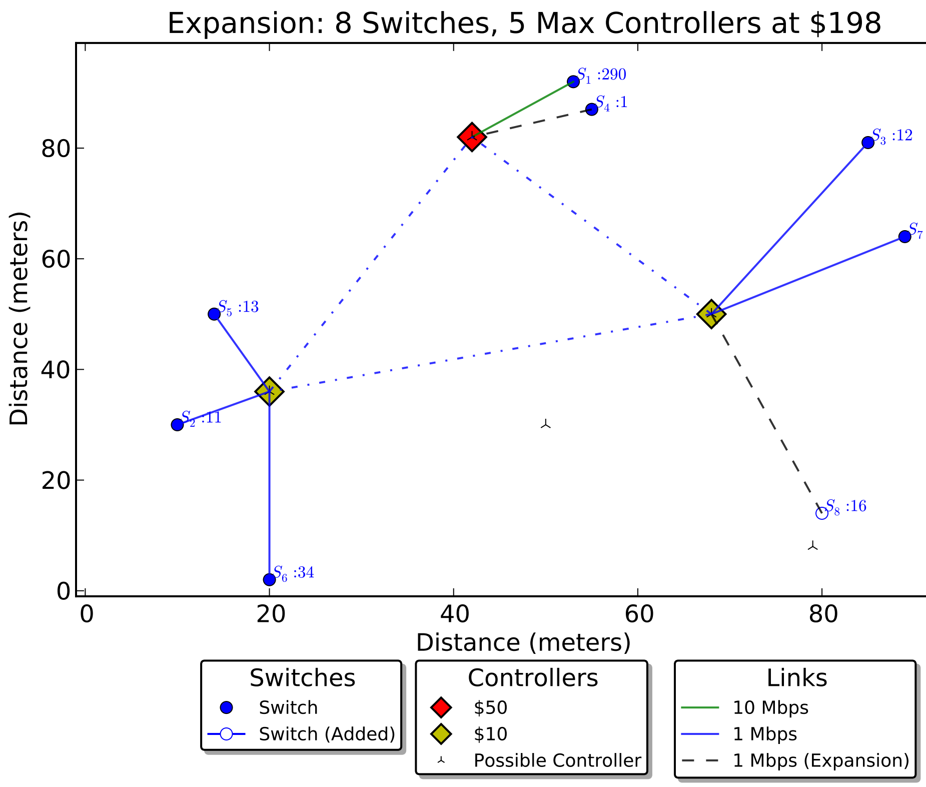


**Figure 4.14:** Optimal solution found by CPLEX for the expansion example.

### 4.2.5 Results for Adding and Removing Switches

Many of the changes that can be made to an existing SDN network is to either add or remove switches. This is the case in a real network where new switches are added to support growth or older switches are removed. In this section, we will optimize using the expansion model by adding or removing switches to an existing network.

Three topologies are generated at random. The switch locations and the possible controller placement locations are generated randomly. For each switch, the number of packets the switch sends are generated randomly between 100 and 999 (as shown in Table 4.7). These topologies are optimized using the same controller and link input parameters as in the planning section (see Table 4.8, and 4.6). The rest of the parameters are the same. The topologies consist of 20, 25 and 30 switches and and for each set of switches, 40 possible controller locations are also generated. The planning model is used to find the optimal solutions for the topologies. The same optimal solutions would have been evident if we used the expansion model to solve the initial topologies but the expansion model generates more variables and extra work has be implemented to handle this scenario. That's why we decided to use the planning model.

Once the planning solutions are found for the 3 topologies, the time and cost of the optimal solutions is recorded. Table 4.14 shows the optimized topologies that we will be using when expanding the networks. The first topology consists of 20 switches and 40 possible placement locations and the optimal solution of $7,746 was found in 176 seconds by the planning model. The second topology has 25 switches with optimal solution cost of $10,810 while reaching the maximum running time. The solution in parenthesis shows that the reported values for topology number 2 are not optimal. The third topology almost reaches the time limit and is by $3,000 less expensive than topology number 2.

Using the results of the planning scenario, we modify each of the topologies and run our expansion model to have a solution for the changed network. First, we optimize by removing existing switches from the network and then by incrementally adding switches in steps of five. When we add switches we also add ten possible placement locations of controllers. Furthermore, each time we add a switch, we generate the location of the switch by randomly generating two integer values in point form $x, y$. With each new switch, the number of packets that it sends to the controller is also

**Table 4.14:** Toplologies that are optimized using the planning model before different expansion scenarios are performed.

| | | | CPLEX | | | | |
|---|---|---|---|---|---|---|---|
| Top | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 20 | 40 | 5 | 30 | 12,190 | 7,746 | 176 |
| 2 | 25 | 40 | (4) | (31) | (14,164) | (10,810) | (108,000) |
| 3 | 30 | 40 | 4 | 36 | 14,278 | 10,909 | 105,636 |

generated randomly between 100 and 999 (see Table 4.7). When expanding the topologies, for each topology we consider that we have the same controller and link inventory as shown in Table 4.8, and 4.6. Figure 4.15 shows topology 3, the planning results are plotted using the optimal solution from the planning model. The 15 added switches (white circles) are plotted. This new topology would be expanded using the expansion model.

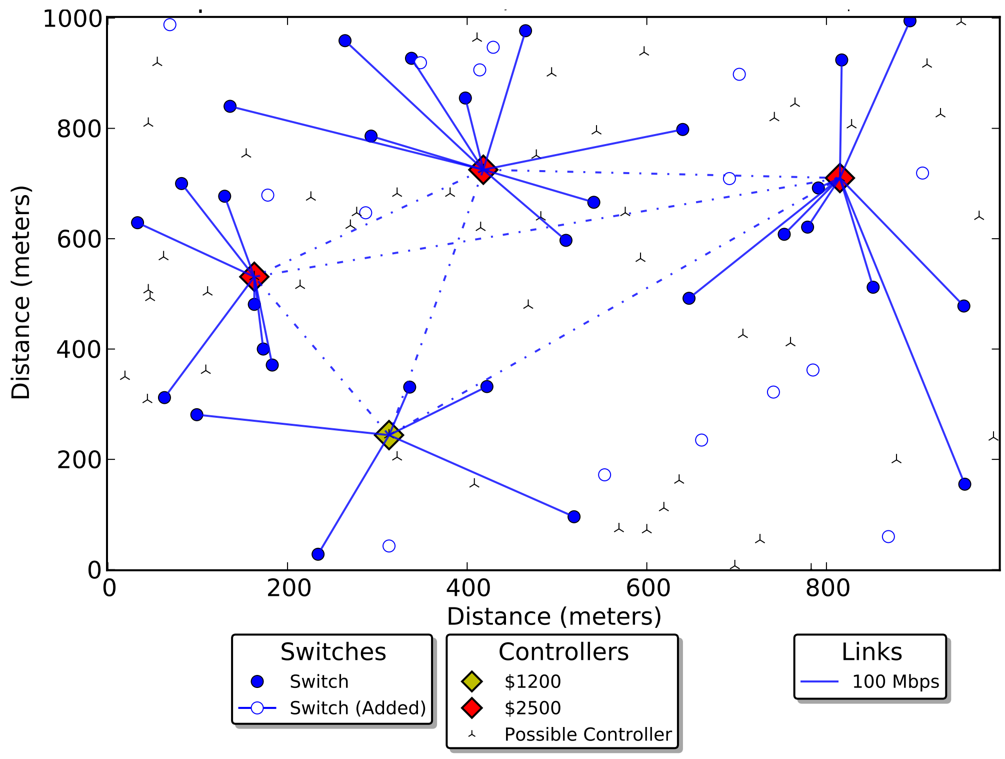


**Figure 4.15:** Existing SDN network with 30 switches and 40 possible controller locations. Fifteen new switches are added and 10 controller placement locations.

The expansion results are shown in Table 4.15. The first column references the starting topology as shown in Table 4.14. The third and fourth columns reflect the changes that are done to the topology by either adding $(+|S|)$ or removing $(-|S|)$ switches. The fifth column $(+|P|)$ shows how many possible placements are added to the input model. The optimal expansion of the network is shown under the column "CPLEX" which shows how many new controllers have been added $(+|C|)$, how many links have been added $(+|L|)$ and how many links have been removed $(-|L|)$. The number of new links includes the links for switches to controllers or inner controller connectivity changes. Finally, the last two columns are the optimal solution (\$) and the time taken to find the optimal solution (CPU).

The first three rows of the expansion results show that 15, 10 and 5 switches were removed from the network. The optimal solution is found instantaneously because the switches are removed and we consider this to be effortless. In this case, we are removing the switches and the links from the switches to the controllers. If in the other scenario where we remove and add a link elsewhere for a switch, here we do take into the consideration the cost to remove the link and place the new link in the optimal solution. Therefore, the cost to remove switches is 0. The next 3 rows for topology one where 5, 10 and 15 switches are added, we have introduced one more controller. If a controller is introduced then it must be connected in a full mesh to other controllers. By examining the planning table, we see that topology one has five controllers already installed. This means that after the expansion, there will be six controllers. We already know that a total of six controllers in a full mesh topology consume 15 links. From, the placement section we know that 10 links are used to connect the controllers. Therefore, it takes five additional links to connect the 6th controller. The other 5 new links connect the 5 new switches that were added. This is why we see 10 new links. Similarly, when we examine the second topology expansion, the solutions to the problems that remove the switches from the network are quick to converge. Also, when 5, 10 and 15 switches are added, only one additional controller is placed. From the previous topology, we have seen that if a controller is added, then additional 5 links are needed. Furthermore, one link from a switch to a controller is removed and added elsewhere and this results of the new links value of 6 links. The third topology is also introducing only one additional controller where an existing switch is reallocated to the new controller.

If we examine the cost of the solutions, we can see that the first topology has the highest cost when adding switches and possible placement locations. This is because a total of 5 links are removed and then added. Removing 5 links multiples the cost of adding by 100 for each removed link. Next, 6 links are removed and this is why we see the cost of 5th row close to the 6th row. As for the second and the third topology, only one link is removed and one controller is added. This gives a constant cost increase because we keep adding five controllers each time. The time taken to find the solutions is considerably less for the expansion problem compared to the placement problem. When the switches are removed, the time taken to find the optimal solution is extremely fast. This is because it is cheaper and faster to remove the links where optimality existed than to recalculate everything. We can see that removing links is less than a second (between 70 to 100 milliseconds).

Figures 4.15 and 4.16 show the result of placement versus expansion for the third topology. There are few items to notice in the figure. As we have seen in the results

**Table 4.15:** Results obtained with CPLEX for expansion scenarios by adding or removing switches.

| | | | | | CPLEX | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Top | $\|S'\|$ | $+\|S\|$ | $-\|S\|$ | $+\|P\|$ | $+\|C\|$ | $+\|L\|$ | $-\|L\|$ | Cost($) | CPU(s) |
| 1 | 5 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 1 | 10 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 1 | 15 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 1 | 25 | 5 | 0 | 10 | 1 | 10 | 5 | 26,183 | 3.67 |
| 1 | 30 | 10 | 0 | 10 | 1 | 16 | 6 | 29,331 | 2.44 |
| 1 | 35 | 15 | 0 | 10 | 1 | 20 | 5 | 32,812 | 1.58 |
| 2 | 10 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 2 | 15 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 2 | 20 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 2 | 30 | 5 | 0 | 10 | 1 | 6 | 1 | 3,563 | $< 1$ |
| 2 | 35 | 10 | 0 | 10 | 1 | 11 | 1 | 7,890 | $< 1$ |
| 2 | 40 | 15 | 0 | 10 | 1 | 17 | 2 | 10,425 | 4.73 |
| 3 | 15 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 3 | 20 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 3 | 25 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | $< 1$ |
| 3 | 35 | 5 | 0 | 10 | 1 | 6 | 1 | 5,529 | $< 1$ |
| 3 | 40 | 10 | 0 | 10 | 1 | 11 | 1 | 10,060 | 1.36 |
| 3 | 45 | 15 | 0 | 10 | 1 | 16 | 1 | 11,880 | 4.60 |

table above, only one link was relocated. This is shown with the solid blue dot and dark dashed lines. The other important observation is that the placement of the new controller is approximately in the middle of the whole network. We can see that some of the new switches that are added during expansion are very far from the new controller which is the only controller connecting them. This is due to the fact that the controller processing capacity does not let any other new switches to be connected to the existing controllers and that changes to existing links is not optimal because of the high cost to remove links.

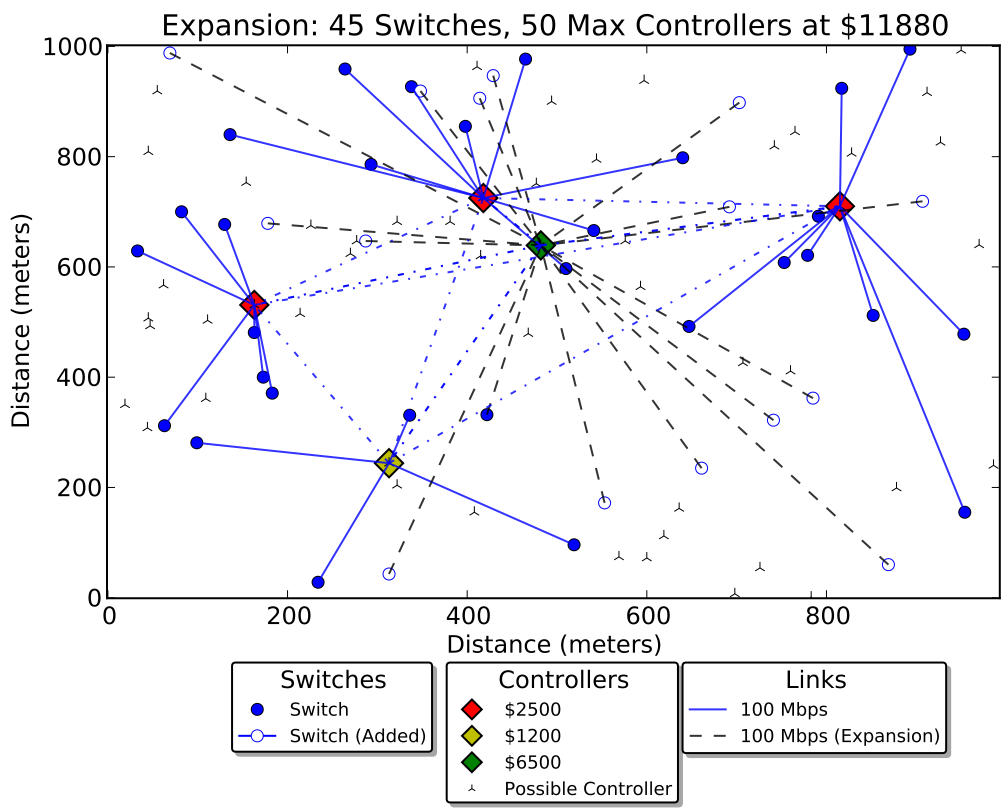


**Figure 4.16:** Expansion to an existing SDN network after adding 15 switches and 10 possible controller locations.

## 4.2.6 Results for Different Cost of Removing Links or Controllers

In Section 4.2.3, we have seen that the optimal placement of the expansion model is influenced by the cost of removing existing items in the network. The more expensive it is to remove an existing link the less likely there will be changes to the existing network. When constraints fail, then the existing network will have to undergo changes. An example may be when an existing link has to transfer more packets than it is physically capable of doing. In this case, the link bandwidth constraint will fail and the existing link will have to be removed and be replaced by a new link. The effect of the removal cost value of the links and controllers will be explored in this section.

The methodology is the same as in Section 4.2.1. Controller and link information is used from the previous section (see Table 4.8, and 4.6). The planning of a topology is done and the planning result is used to indicate which of the controllers and links already exist in the expansion model. Twenty switches and 40 possible controller locations are generated for the topology and the optimizer finds the solution. Table 4.16 shows the optimization result for the topology. The solution time has exceeded the 30 hours time limit and the optimizer is stopped. This means that a better optimal cost for this solution may exist. The topology is depicted in Figure 4.17.

**Table 4.16:** A topology that is optimized before different expansion scenarios are run to add 10 switches with multiple cost variations.

| | | CPLEX | | | | |
|---|---|---|---|---|---|---|
| $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 20 | 40 | (4) | (26) | (10,155) | (7,696) | 108,000 |

The results to the expansion of networks with different cost to remove items from the network is optimized and shown in Table 4.17. The first column is the problem number and the second column is the Cost Multiplier. The third column ($+|S|$) shows how many switches were added to the network. Since the controller possible placement locations are 40, it was not necessary to add more locations because only 4 placement locations were used in the planning model. The remainder of the results show the number of controllers that were added or changed ($+|C|$), the number of links added ($+|L|$) and the number of links that were removed ($-|L|$).

The cost multiplier starts with a very high value where if a link of 5 meters costs $6 to plan, the removal of it for the first problem would be $600. Down until the 5th problem, the removal of a link would cost more than the planning. Problem number 6 has a removal cost that is the same as the planning cost. Finally, problems 7 and 8 make it possible for the existing network to change because the cost to move a link from one place to another is $3 and $0.06 (compared to $6 planning cost).

CPLEX results show that throughout the different problems, at least one controller is added ($+|C|$). The minimum number of links that are added are 10 because we are adding 10 switches to the network. Problems 1, 2 and 3 all have the same cost because one controller is added and all the new switches are connected. Problem

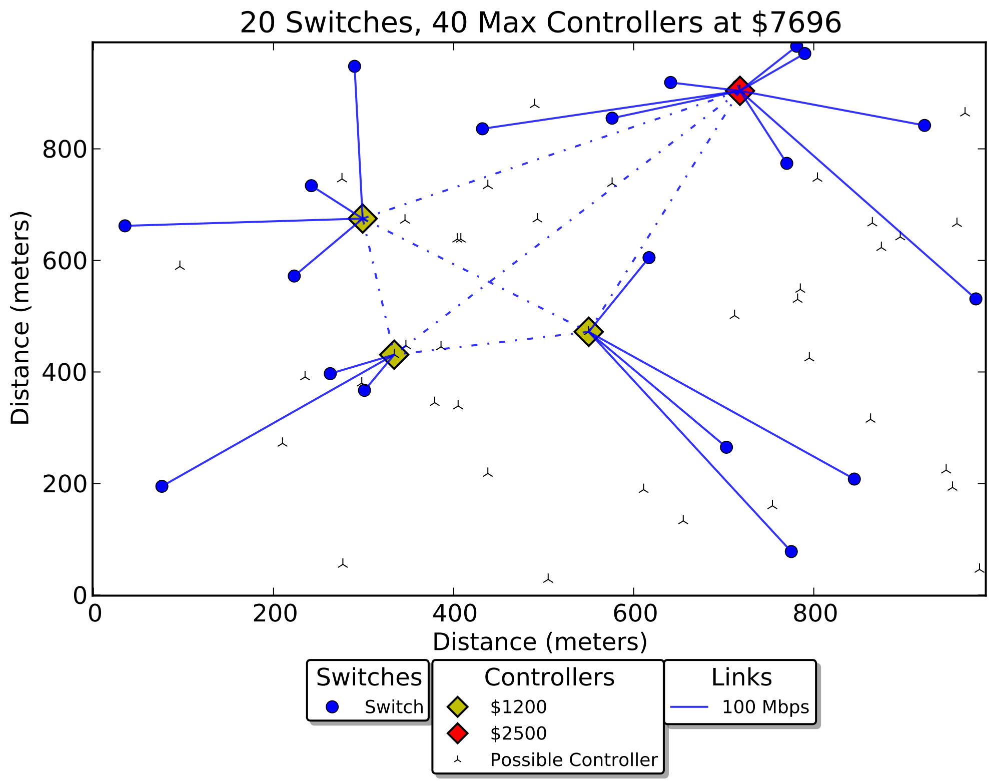


**Figure 4.17:** Existing SDN network with 20 switches and 40 possible controller locations.

number 4 adds two controllers and removes 4 links. The 4 links that are removed, are then added back which is why we see 14 new links on the result table. The optimal cost of the solution of problem 4 is less compared to problem number 3 because the solver found it cheaper to reallocate 4 links ($-|L|$). This is because the cost multiplier decreased by 50% from problem 3 to problem 4. Problem number 5, 6, 7 and 8 all place two types of controllers and reallocate 2 links. The interesting part here is that although the number of switches and the number of links remain the same, the optimal solution price gets cheaper for every problem. Problem 8 is cheaper than problem 7, problem 7 is cheaper than problem 6 and problem 6 is cheaper than problem 5. This happens because as the problem number increases, the cost multiplier decreases. And it will affect the optimal price of the solutions because the same number of links is being removed and then added again elsewhere (for problems 5 to 8). The results from problems 1 to 3 are verified because the cost remains the same even though

**Table 4.17:** Results obtained with CPLEX for expansion scenarios with multiple cost variations.

| | | | CPLEX | | | | |
|---|---|---|---|---|---|---|---|
| Problem | Cost Multiplier | $\|+S\|$ | $+\|C\|$ | $+\|L\|$ | $-\|L\|$ | Cost($) | CPU(s) |
| 1 | 100 | 10 | 1 | 10 | 0 | 7,511 | 1 |
| 2 | 80 | 10 | 1 | 10 | 0 | 7,511 | <1 |
| 3 | 50 | 10 | 1 | 10 | 0 | 7,511 | 2 |
| 4 | 25 | 10 | 2 | 14 | 4 | 7,470 | 11 |
| 5 | 10 | 10 | 2 | 12 | 2 | 6,101 | 8 |
| 6 | 1 | 10 | 2 | 12 | 2 | 5,171 | 25 |
| 7 | 0.5 | 10 | 2 | 12 | 2 | 5,120 | 21 |
| 8 | 0.1 | 10 | 2 | 12 | 2 | 5,078 | 84 |

the cost multiplier is decreasing. The cost multiplier only affects the items that are removed and problems 1 to 3 do not have any links that are removed so the optimal solution does not change. There is a pattern in finding the optimal solution for every problem where, as the cost multiplier decreases, the time taken to find the optimal solution increases.

Figure 4.18 shows the optimal expansion of problem number 8. The figure shows two switches that changed the controller that they were connected to from the planning phase. The dashed black coloured links show which link were reallocated during the expansion. A controller of type two and type one is also added by the end of the expansion.

## 4.3 Concluding Remarks

Based on the optimizations results of different scenarios for the planning and expansion model, we were able to understand how the models perform in terms of time taken and cost to find the optimal solution. The results from the planning model tells us how the planning model performs and the results from the expansion model show us what can be done to change an existing network and finally shows the cost and time taken to find the optimal solution.

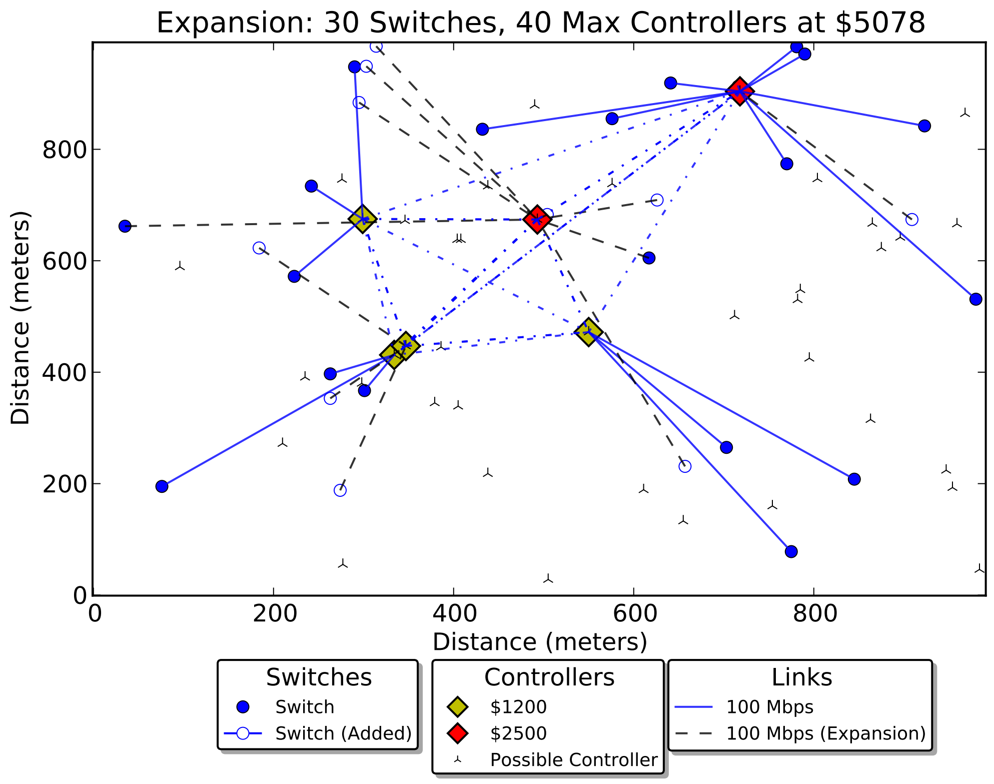


**Figure 4.18:** The expansion to existing SDN network by setting the costs of removing items to 1/10th of what planning costs are.

The planning result starts from a small input size and grows to a large input size to see how the model performs. The planning model is used to optimize 36 different problems. The first problem, at sample size of 10 switches and 5 possible controller locations, the time taken to find the optimal solution is less than a second. However, the last problem, at an input size of 200 switches and 20 possible controller locations, the time taken to find an optimal solution is over 30 hours. The size of the problem is increased from the small problem size to the large problem size and its cost and time to find the solution is recorded. We have seen a pattern for the solution time that, as the problem size increased, the time to find the optimal solution also increased in non-linear form. Furthermore, the price of the solution always increased with the input size. This is expected as the problem type falls in the NP-Complete. We have also investigated at what happens to the cost of the solution as we increased the possible controller locations while keeping the number of switches the same. We found that the more controller locations that are present, the cheaper the solution cost became (up to a certain point).

First, expansion result starts by expanding three topologies where switches are either added or removed. Second, we investigated how the cost to remove items from an existing SDN network affects the expansion result. The expansion result to remove switches from an existing network shows that it is a no cost step and it is always a quick operation. However, when switches are added, to expand the network, the solver takes time but it is a quick operation. The existing switches are either reallocated to another controller or they remain the same. As for the new switches, most of the time they are connected to a newly placed controller. Finally, the controller connectivity is established with the new controller(s) and the rest of the existing controller(s). We repeatedly investigated the result of the expansion by having an expansion cost that started at 100 times the planning cost and decreasing it until $\frac{1}{10}$ of the planning cost. At a cost of 100 times the planning cost to remove existing items from the network, adding 10 switches did not make any changes to the existing network. Only an additional controller is added to support the 10 switches. The time taken to expand the problem was within 1 second. As the cost to remove existing items from the network decreased, the optimal solution cost for the expansion also decreased. Furthermore, changes to the existing network is suggested and the time to find the optimal solutions increased. The next chapter summarizes the thesis and makes recommendations for improvements.

# Chapter 5

# Conclusions and Future Work

Since SDN networks move the decision making into a controller, it is essential to develop methods to find the optimal placement of the controller(s) in a network. This study finds the optimal placement of controllers on a network by suggesting the location and the number of controllers for two network planning scenarios: a new SDN network and an existing SDN network that requires change. Moving current networks to SDN is optimized by a mathematical model that minimizes the cost of placing controllers. The objective is to find the minimum cost of placing the controllers while considering network latency for flow setup, controller processing, link bandwidth controllers and switches and finally, connectivity between controllers. Existing SDN networks that need planning process are modelled by the second mathematical model that considers already installed equipment.

The planning model consists of a binary integer program that minimizes the cost of placing controllers on a network while keeping the flow setup latency under a threshold and respecting the inventory that is available for the planning process. The model also takes into consideration the link bandwidth between switches and controllers in the case full flows are sent to the controller for processing. In the event that multiple controllers are installed, they can then be connected to each other using the full mesh topology. The expansion model is a mixed integer program and suggests changes to the existing SDN network by performing the planning process over the existing infrastructure. For every part of the existing network, the expansion model decides if it is better to remove the existing infrastructure, replace it with a better option from the inventory or just add new equipment.

Prior to planning, different types of network topologies are generated that take into account profiles of controllers and links, randomized locations of switches and randomized possible placement controller locations. Results indicate that optimal controller planning is performed in under one second when input size is small (i.e. 10 switches). In general, we can see a pattern that shows that as the input size increases, the time to find the optimal solution also increases. For example, when the number of switches is increased from 10 to 20, it takes 13 times longer to find the optimal solution.

In terms of cost, we can also see some general trends. The first thing we can observe is that the cost of solutions is higher as the input size increases. As the number of possible controller locations increased, for the same number of switches, the optimal cost was found to be slightly cheaper (up to a certain point). For example, planning 100 switches gives an optimal cost of \$49,489 for 10 possible controller placement locations. Increasing the number of possible controller locations to 20, brings the optimal cost of placing controllers to \$47,172. The optimal cost is cheaper because the optimizer is able to place controllers closer to the switches.

Our expansion includes two types of experiments. The first investigates the result of the expansion model by adding and removing switches. The second experiment investigates how the existing network changes with different cost to remove existing inventory from the network. Before any expansion topology can be optimized, planning the network by the planning model must be done. For the first experiment, 3 different networks are planned randomly that form the topologies that our expansion model performs modifications on. Then for each of the topologies, 15, 10, and 5 switches are removed from the network and 5, 10, and 15 switches are added to the network before expansion is run. The results of the optimal solutions show that removing items from the network is inexpensive in cost and time. However, when adding 5, 10 and 15 switches to the network, the solution time and cost of the network increases. However, when switches are added, to expand the network, the solver takes time but it is a quick operation. The time to expand an existing network by adding 15 switches ranges from 1 to 4 seconds. The cost of the solutions depend on wether an existing item is reallocated, if so, the optimal solution cost is much more expensive.

The second expansion experiment involves in adding a set of switches to an existing topology with the cost to remove the existing inventory that starts at 100 times the planning cost and goes down to $\frac{1}{10}$ of the planning cost. From the results, we observe that the optimal solution when the removal cost of the existing inventory is

100, the existing network is not changed at all. A controller is placed for the additional switches that were added. As the cost to remove existing inventory decreased, the existing network started changing and the optimal solution cost also decreased. Furthermore, the cost to remove existing inventory decreased, changes to the existing network became more frequent.

## 5.1 Future Work

We have limited our flow setup latencies because we are not including queuing on controllers that process switch requests. We only consider the transmission, propagation and processing delays. Adding queuing delay could be an improvement because a switch would be guaranteed a maximum end-to-end delay.

Currently, before we generate our expansion model, in our implementation, we can only make a few changes to the vectors of our input to generate the model files. For example, only removing or adding of switches or placement locations is possible. Another improvement could be to analyze the vectors of the switches and possible controller locations and, based on the changes that are proposed for the expansion, the implementation would reorganize the existing controller and switch configurations and generate a proper expansion model.

When the topology is formed, we assume that only table synchronization between controllers is performed. This may not always be the case because a controller may be configured to send traffic to the next possible controller for inspection (in case of a deep packet inspection for example). With our assumption, we have limited our models to the cheapest link type when connecting the controllers. A future improvement to the models would be to pick the correct link type between controllers.

The exact algorithm takes a lot of time to find the optimal solutions. In our examples, we have seen that it takes 19 days for 4 instances to run the planning of controllers on a SDN network. One solution to have improvements over the exact method is to run many problems and allow the solver to auto tune the parameters for each problem. This could reveal parameters that could help with future optimizations. Even so, the improvement would be negligible because we already know that the planning and expansion problems are NP-Complete and that the complexity increases with respect to the problem size. A possible improvement is to derive a local search algorithm that would find a solution in polynomial time. However, there is no guarantee that it will find the optimal solution because the algorithm may become stuck in the first local minimum that it encounters. As a result, future work can be expanded to apply more advanced heuristic algorithms such as Tabu Search. This would help the algorithm pass the local minimum and find solutions that are closer to the optimal solution [47].

# List of References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and control element separation (forces) framework," *RFC3746*, pp. 5–30, 2004.

[3] M. Kind, F. Westphal, A. Gladisch, and S. Topp, "Splitarchitecture: Applying the software defined networking concept to carrier networks," in *World Telecommunications Congress (WTC), 2012*, pp. 1–6, 2012.

[4] L. Li, Z. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 7–12, 2012.

[5] D. Venmani, D. Zeghlache, and Y. Gourhant, "Demystifying link congestion in 4g-lte backhaul using openflow," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pp. 1–8, 2012.

[6] T. Luo, H.-P. Tan, and T. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *Communications Letters, IEEE*, vol. 16, no. 11, pp. 1896–1899, 2012.

[7] D. Simeonidou, R. Nejabati, and S. Azodolmolky, "Enabling the future optical internet with openflow: A paradigm shift in providing intelligent optical network services," in *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, pp. 1–4, 2011.

[8] S. Gringeri, N. Bitar, and T. Xia, "Extending software defined network principles to include optical transport," *Communications Magazine, IEEE*, vol. 51, no. 3, pp. 32–40, 2013.

[9] M. Shirazipour, Y. Zhang, N. Beheshti, G. Lefebvre, and M. Tatipamula, "Openflow and multi-layer extensions: Overview and next steps," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 13–17, 2012.

[10] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, "Packet and circuit network convergence with openflow," in *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, pp. 1–3, 2010.

[11] V. Gudla, S. Das, A. Shastri, G. Parulkar, N. McKeown, L. Kazovsky, and S. Yamashita, "Experimental demonstration of openflow control of packet and circuit switches," in *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, pp. 1–3, 2010.

[12] M. Shirazipour, W. John, J. Kempf, H. Green, and M. Tatipamula, "Realizing packet-optical integration with sdn and openflow 1.1 extensions," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 6633–6637, IEEE, 2012.

[13] M. Channegowda, P. Kostecki, N. Efstathiou, S. Azodolmolky, R. Nejabati, P. Kaczmarek, A. Autenrieth, J.-P. Elbers, and D. Simeonidou, "Experimental evaluation of extended openflow deployment for high-performance optical networks," in *European Conference and Exhibition on Optical Communication*, Optical Society of America, 2012.

[14] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "Where is the debugger for my software-defined network?," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 55–60, ACM, 2012.

[15] M. Canini and D. Kostic, "Systematic software testing meets networking,"

[16] M.-K. Shin, H. H. Kwak, J.-Y. Choi, and M. Kang, "Verisdn: Formal verification for software-defined networking (sdn),"

[17] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.

[18] P. H. V. Guimaraes, L. H. G. Ferraz, J. V. Torres, I. D. Alvarenga, C. S. Rodrigues, and O. C. M. Duarte, "Experimenting content-centric networks in the future internet testbed environment," in *Workshop on Cloud Convergence, ICC*, 2013.

[19] M. Kuzniar, M. Canini, and D. Kostic, "Often testing openflow networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 54–60, IEEE, 2012.

[20] V. Mann, A. Vishnoi, A. Iyer, and P. Bhattacharya, "Vmpatrol: Dynamic and automated qos for virtual machine migrations," in *Network and Service Management (CNSM), 2012 8th International Conference on*, pp. 174–178, IEEE, 2012.

[21] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, "Live migration of an entire network (and its hosts)," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 109–114, ACM, 2012.

[22] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "Vertigo: Network virtualization and beyond," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 24–29, IEEE, 2012.

[23] Z. Bozakov, "An open router virtualization framework using a programmable forwarding plane," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 439–440, 2010.

[24] Z. Bozakov, "Architecture and algorithms for virtual routers as a service," in *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, pp. 1–3, IEEE, 2011.

[25] M. Soliman, B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Source routed forwarding with software defined control, considerations and implications," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pp. 43–44, ACM, 2012.

[26] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[27] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," *Proc. INM/WREN*, vol. 10, pp. 1–1, 2010.

[28] K. Nam-Seok and H. Hwanjo, "Openqflow: Scalable openflow with flow-based qos," *IEICE transactions on communications*, vol. 96, no. 2, pp. 479–488, 2013.

[29] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, and S. Civanlar, "Scalable video streaming over openflow networks: An optimization framework for qos routing," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 2241–2244, IEEE, 2011.

[30] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for qos-enabled adaptive video streaming over openflow networks," 2013.

[31] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[32] "Trema openflow controller," Aug 2013.

[33] D. Erickson, "Floodlight java based openflow controller," *Last accessed, Ago*, 2012.

[34] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2012.

[35] A. Voellmy and J. Wang, "Scalable software defined network controllers," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 289–290, ACM, 2012.

[36] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, *et al.*, "Onix: A distributed control platform for large-scale production networks.," in *OSDI*, vol. 10, pp. 1–6, 2010.

[37] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," 2013.

[38] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, (New York, NY, USA), pp. 7–12, ACM, 2012.

[39] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR) 19th International Conference in Pattern Recognition (ICPR).

[40] Y. nan HU, W. dong WANG, X. yang GONG, X. rong QUE, and S. duan CHENG, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, Supplement 2, no. 0, pp. 92 – 171, 2012.

[41] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," *International Conference on Network and Service Management*, p. 188, May 2013.

[42] W. L. Winston and J. B. Goldberg, *Operations research: applications and algorithms*. Thomson/Brooks/Cole Belmont, 2004.

[43] M. R. Pasandideh and M. St-Hilaire, "Automatic planning of 3g umts all-ip release 4 networks with realistic traffic," *Computers & Operations Research*, vol. 40, no. 8, pp. 1991–2003, 2013.

[44] S. Chamberland, M. St-Hilaire, and S. Pierre, "An analysis of different colocated router network topologies within a pop in ip networks," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 2, pp. 733–736 vol.2, 2003.

[45] I. I. CPLEX, "V12. 1: Users manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[46] R. G. Michael and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *WH Freeman & Co., San Francisco*, 1979.

[47] M. Sun, "A tabu search heuristic for the uncapacitated facility location problem," in *Metaheuristic Optimization via Memory and Evolution* (R. Sharda, S. Vo, C. Rego, and B. Alidaee, eds.), vol. 30 of *Operations Research/Computer Science Interfaces Series*, pp. 191–211, Springer US, 2005.

# Appendix A

# Small to Large Input Sizes Results

## A.1 Instance 1

**Table A.1:** Small to Large Input Sizes: Instance 1

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\lvert S\rvert$ | $\lvert P\rvert$ | $\lvert P'\rvert$ | $\lvert L'\rvert$ | Packets | Cost($) | CPU(s) |
| 1 | 10 | 5 | 2 | 11 | 5,350 | 3,013 | $< 1$ |
| 2 | 10 | 10 | 3 | 13 | 6,124 | 4,450 | $< 1$ |
| 3 | 10 | 15 | 3 | 13 | 5,196 | 4,458 | 5 |
| 4 | 10 | 20 | 3 | 13 | 5,584 | 4,398 | 8 |
| 5 | 20 | 5 | 1 | 20 | 9,972 | 4,622 | $< 1$ |
| 6 | 20 | 10 | 5 | 30 | 11,537 | 8,905 | 7 |
| 7 | 20 | 15 | 4 | 26 | 9,149 | 6,592 | 5 |
| 8 | 20 | 20 | 4 | 26 | 11,237 | 7,964 | 79 |
| 9 | 30 | 5 | 1 | 30 | 14,617 | 6,679 | $< 1$ |
| 10 | 30 | 10 | 5 | 40 | 16,531 | 12,880 | 1 |
| 11 | 30 | 15 | 6 | 45 | 17,325 | 13,179 | 150 |
| 12 | 30 | 20 | 5 | 40 | 16,362 | 12,388 | 3,195 |
| 13 | 40 | 5 | 2 | 41 | 20,143 | 8,254 | $< 1$ |
| 14 | 40 | 10 | 6 | 55 | 24,388 | 20,566 | 42.62 |
| 15 | 40 | 15 | 6 | 55 | 21,117 | 17,415 | 2,614 |
| 16 | 40 | 20 | 7 | 61 | 25,216 | 20,227 | 11,388 |
| 17 | 50 | 5 | 2 | 51 | 24,142 | 8,706 | $< 1$ |
| 18 | 50 | 10 | 5 | 60 | 27,060 | 23,631 | 6.04 |
| 19 | 50 | 15 | 5 | 60 | 29,491 | 25,497 | 15,834 |
| 20 | 50 | 20 | 7 | 71 | 27,911 | 21,896 | 1,185 |
| 21 | 75 | 5 | 3 | 78 | 44,172 | 13,904 | $< 1$ |
| 22 | 75 | 10 | 8 | 103 | 41,493 | 36,325 | 378 |
| 23 | 75 | 15 | 8 | 103 | 41,054 | 36,493 | 6,967 |
| 24 | 75 | 20 | 6 | 90 | 40,673 | 35,868 | 12,375 |
| 25 | 100 | 5 | 4 | 106 | 56,461 | 16,480 | $< 1$ |
| 26 | 100 | 10 | 9 | 136 | 56,143 | 49,989 | 32 |
| 27 | 100 | 15 | 11 | 155 | 54,528 | 48,059 | 52,280 |
| 28 | 100 | 20 | 9 | 136 | 57,395 | 49,634 | 1,068 |
| 29 | 150 | 5 | 3 | 153 | 86,751 | 25,599 | $< 1$ |
| 30 | 150 | 10 | 7 | 171 | 82,911 | 75,107 | 35 |
| 31 | 150 | 15 | 13 | 228 | 84,054 | 75,458 | 65,546 |
| 32 | 150 | 20 | 8 | 178 | 85,010 | 77,531 | 108,000 |
| 33 | 200 | 5 | 5 | 210 | 113,006 | 32,612 | $< 1$ |
| 34 | 200 | 10 | 10 | 245 | 115,517 | 109,004 | 96 |
| 35 | 200 | 15 | 10 | 245 | 104,323 | 95,112 | 7,310 |
| 36 | 200 | 20 | 11 | 255 | 106,902 | 100,031 | 108,000 |

## A.2 Instance 2

**Table A.2:** Small to Large Input Sizes: Instance 2

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\lvert S\rvert$ | $\lvert P\rvert$ | $\lvert P'\rvert$ | $\lvert L'\rvert$ | Packets | Cost($) | CPU(s) |
| 1 | 10 | 5 | 2 | 11 | 5,350 | 3,012 | $< 1$ |
| 2 | 10 | 10 | 3 | 13 | 6,124 | 4,450 | $< 1$ |
| 3 | 10 | 15 | 3 | 13 | 5,196 | 4,458 | 5 |
| 4 | 10 | 20 | 3 | 13 | 5,584 | 4,398 | 8 |
| 5 | 20 | 5 | 1 | 20 | 9,972 | 4,622 | $< 1$ |
| 6 | 20 | 10 | 5 | 30 | 11,537 | 8,905 | 7 |
| 7 | 20 | 15 | 4 | 26 | 9,149 | 6,592 | 5 |
| 8 | 20 | 20 | 4 | 26 | 11,237 | 7,964 | 79 |
| 9 | 30 | 5 | 1 | 30 | 14,617 | 6,678 | $< 1$ |
| 10 | 30 | 10 | 5 | 40 | 16,531 | 12,880 | 1 |
| 11 | 30 | 15 | 6 | 45 | 17,325 | 13,179 | 150 |
| 12 | 30 | 20 | 5 | 40 | 16,362 | 12,388 | 3,195 |
| 13 | 40 | 5 | 2 | 41 | 20,143 | 8,253 | $< 1$ |
| 14 | 40 | 10 | 6 | 55 | 24,388 | 20,566 | 42.62 |
| 15 | 40 | 15 | 6 | 55 | 21,117 | 17,415 | 2,614 |
| 16 | 40 | 20 | 7 | 61 | 25,216 | 20,227 | 11,388 |
| 17 | 50 | 5 | 2 | 51 | 24,142 | 8,705 | $< 1$ |
| 18 | 50 | 10 | 5 | 60 | 27,060 | 23,631 | 6.04 |
| 19 | 50 | 15 | 5 | 60 | 29,491 | 25,497 | 15,834 |
| 20 | 50 | 20 | 7 | 71 | 27,911 | 21,896 | 1,185 |
| 21 | 75 | 5 | 3 | 78 | 44,172 | 13,904 | $< 1$ |
| 22 | 75 | 10 | 8 | 103 | 41,493 | 36,325 | 378 |
| 23 | 75 | 15 | 8 | 103 | 41,054 | 36,493 | 6,967 |
| 24 | 75 | 20 | 6 | 90 | 40,673 | 35,868 | 12,375 |
| 25 | 100 | 5 | 4 | 106 | 56,461 | 16,480 | $< 1$ |
| 26 | 100 | 10 | 9 | 136 | 56,143 | 49,989 | 32 |
| 27 | 100 | 15 | 11 | 155 | 54,528 | 48,059 | 52,280 |
| 28 | 100 | 20 | 9 | 136 | 57,395 | 49,634 | 1,068 |
| 29 | 150 | 5 | 3 | 153 | 86,751 | 25,598 | $< 1$ |
| 30 | 150 | 10 | 7 | 171 | 82,911 | 75,107 | 35 |
| 31 | 150 | 15 | 13 | 228 | 84,054 | 75,458 | 65,546 |
| 32 | 150 | 20 | 8 | 178 | 85,010 | 77,531 | 108,000 |
| 33 | 200 | 5 | 5 | 210 | 113,006 | 32,612 | $< 1$ |
| 34 | 200 | 10 | 10 | 245 | 115,517 | 109,004 | 96 |
| 35 | 200 | 15 | 10 | 245 | 104,323 | 95,112 | 7,310 |
| 36 | 200 | 20 | 11 | 255 | 106,902 | 100,031 | 108,000 |

## A.3 Instance 3

**Table A.3:** Small to Large Input Sizes: Instance 3

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 10 | 5 | 2 | 11 | 5,122 | 3,171 | $< 1$ |
| 2 | 10 | 10 | 3 | 13 | 5,717 | 4,460 | $< 1$ |
| 3 | 10 | 15 | 2 | 11 | 4,741 | 3,240 | $< 1$ |
| 4 | 10 | 20 | 2 | 11 | 4,702 | 3,163 | $< 1$ |
| 5 | 20 | 5 | 1 | 20 | 11,072 | 4,629 | $< 1$ |
| 6 | 20 | 10 | 4 | 26 | 9,782 | 6,820 | $< 1$ |
| 7 | 20 | 15 | 3 | 23 | 10,339 | 8,015 | 69 |
| 8 | 20 | 20 | 4 | 26 | 10,260 | 7,726 | 80 |
| 9 | 30 | 5 | 1 | 30 | 16,792 | 5,655 | $< 1$ |
| 10 | 30 | 10 | 4 | 36 | 15,750 | 12,209 | 21 |
| 11 | 30 | 15 | 6 | 45 | 17,036 | 12,801 | 118 |
| 12 | 30 | 20 | 6 | 45 | 17,249 | 13,141 | 363 |
| 13 | 40 | 5 | 3 | 43 | 21,512 | 8,278 | $< 1$ |
| 14 | 40 | 10 | 6 | 55 | 21,645 | 17,322 | 25 |
| 15 | 40 | 15 | 6 | 55 | 23,622 | 18,736 | 103 |
| 16 | 40 | 20 | 6 | 55 | 23,813 | 18,692 | 1,405 |
| 17 | 50 | 5 | 2 | 51 | 28,126 | 8,675 | $< 1$ |
| 18 | 50 | 10 | 6 | 65 | 28,621 | 24,204 | 17 |
| 19 | 50 | 15 | 8 | 78 | 28,811 | 23,515 | 578 |
| 20 | 50 | 20 | 7 | 71 | 27,186 | 22,632 | 376 |
| 21 | 75 | 5 | 3 | 78 | 37,457 | 12,770 | $< 1$ |
| 22 | 75 | 10 | 10 | 120 | 43,170 | 38,400 | 452 |
| 23 | 75 | 15 | 7 | 96 | 38,346 | 33,109 | 304 |
| 24 | 75 | 20 | 7 | 96 | 40,088 | 35,129 | 108,000 |
| 25 | 100 | 5 | 4 | 106 | 56,192 | 17,813 | $< 1$ |
| 26 | 100 | 10 | 8 | 128 | 52,002 | 47,092 | 95 |
| 27 | 100 | 15 | 12 | 166 | 58,335 | 50,520 | 1,992 |
| 28 | 100 | 20 | 13 | 178 | 51,515 | 45,282 | 32,221 |
| 29 | 150 | 5 | 3 | 153 | 86,917 | 27,695 | $< 1$ |
| 30 | 150 | 10 | 7 | 171 | 84,813 | 77,693 | 10,147 |
| 31 | 150 | 15 | 12 | 216 | 77,743 | 70,719 | 108,000 |
| 32 | 150 | 20 | 12 | 216 | 78,131 | 70,464 | 108,000 |
| 33 | 200 | 5 | 5 | 210 | 110,658 | 34,392 | $< 1$ |
| 34 | 200 | 10 | 10 | 245 | 107,880 | 103,052 | 87 |
| 35 | 200 | 15 | 11 | 255 | 111,720 | 103,721 | 26,496 |
| 36 | 200 | 20 | 11 | 255 | 108,053 | 98,300 | 108,000 |

## A.4 Instance 4

**Table A.4:** Small to Large Input Sizes: Instance 4

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 10 | 5 | 2 | 11 | 5,209 | 3,544 | $< 1$ |
| 2 | 10 | 10 | 2 | 11 | 4,973 | 3,517 | $< 1$ |
| 3 | 10 | 15 | 2 | 11 | 4,897 | 3,139 | $< 1$ |
| 4 | 10 | 20 | 3 | 13 | 5,157 | 4,488 | 10.26 |
| 5 | 20 | 5 | 1 | 20 | 11,974 | 4,419 | $< 1$ |
| 6 | 20 | 10 | 5 | 30 | 11,745 | 8,341 | 1 |
| 7 | 20 | 15 | 4 | 26 | 10,770 | 7,777 | 32 |
| 8 | 20 | 20 | 3 | 23 | 10,267 | 7,710 | 151 |
| 9 | 30 | 5 | 1 | 30 | 16,525 | 5,435 | $< 1$ |
| 10 | 30 | 10 | 6 | 45 | 17,271 | 13,315 | 16 |
| 11 | 30 | 15 | 6 | 45 | 17,462 | 13,384 | 241 |
| 12 | 30 | 20 | 6 | 45 | 17,697 | 13,173 | 236 |
| 13 | 40 | 5 | 3 | 43 | 21,789 | 8,167 | $< 1$ |
| 14 | 40 | 10 | 6 | 55 | 24,068 | 20,429 | 89 |
| 15 | 40 | 15 | 6 | 55 | 20,457 | 16,325 | 107 |
| 16 | 40 | 20 | 6 | 55 | 21,924 | 17,245 | 935 |
| 17 | 50 | 5 | 2 | 51 | 25,672 | 8,969 | $< 1$ |
| 18 | 50 | 10 | 6 | 65 | 28,351 | 24,281 | 84 |
| 19 | 50 | 15 | 7 | 71 | 26,971 | 21,710 | 117 |
| 20 | 50 | 20 | 7 | 71 | 26,871 | 21,539 | 2,225 |
| 21 | 75 | 5 | 3 | 78 | 40,761 | 12,725 | $< 1$ |
| 22 | 75 | 10 | 6 | 90 | 43,216 | 38,427 | 165 |
| 23 | 75 | 15 | 8 | 103 | 41,050 | 35,693 | 5,115 |
| 24 | 75 | 20 | 8 | 103 | 41,932 | 35,294 | 6,251 |
| 25 | 100 | 5 | 4 | 106 | 56,747 | 18,563 | $< 1$ |
| 26 | 100 | 10 | 10 | 145 | 54,705 | 48,399 | 1,721 |
| 27 | 100 | 15 | 8 | 128 | 53,919 | 46,823 | 1,194 |
| 28 | 100 | 20 | 11 | 155 | 53,442 | 46,656 | 33,368 |
| 29 | 150 | 5 | 3 | 153 | 82,763 | 26,679 | $< 1$ |
| 30 | 150 | 10 | 8 | 178 | 74,870 | 68,263 | 225 |
| 31 | 150 | 15 | 10 | 195 | 82,684 | 74,151 | 5,998 |
| 32 | 150 | 20 | 14 | 241 | 80,995 | 72,670 | 108,000 |
| 33 | 200 | 5 | 4 | 206 | 111,199 | 34,271 | $< 1$ |
| 34 | 200 | 10 | 10 | 245 | 108,189 | 102,077 | 23.08 |
| 35 | 200 | 15 | 12 | 266 | 114,488 | 105,961 | 108,000 |
| 36 | 200 | 20 | 12 | 266 | 106,630 | 96,696 | 108,000 |

# Appendix B

# Increasing Possible Controller Locations Results

## B.1 Instance 1

Table B.1: Increasing Possible Controller Locations: Instance 1

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 20 | 5 | 5 | 30 | 11,524 | 9,090 | $< 1$ |
| 2 | 20 | 10 | 4 | 26 | 10,985 | 7,866 | 3 |
| 3 | 20 | 15 | 4 | 26 | 10,652 | 7,933 | 17 |
| 4 | 20 | 20 | 5 | 30 | 11,915 | 8,259 | 85 |
| 5 | 20 | 25 | 3 | 23 | 10,220 | 7,693 | 857 |
| 6 | 20 | 30 | 3 | 23 | 10,150 | 7,652 | 4,396 |
| 7 | 20 | 40 | 5 | 30 | 12,240 | 7,875 | 244 |
| 8 | 20 | 50 | 3 | 23 | 8,945 | 6,344 | 628 |
| 9 | 20 | 75 | 4 | 26 | 9,951 | 6,779 | 5,621 |

## B.2 Instance 2

**Table B.2:** Increasing Possible Controller Locations: Instance 2

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 20 | 5 | 4 | 26 | 10,887 | 12,037 | $< 1$ |
| 2 | 20 | 10 | 5 | 30 | 12,201 | 13,289 | $< 1$ |
| 3 | 20 | 15 | 4 | 26 | 10,442 | 10,347 | 7 |
| 4 | 20 | 20 | 4 | 26 | 11,251 | 11,687 | 336 |
| 5 | 20 | 25 | 5 | 30 | 14,250 | 14,465 | 1,123 |
| 6 | 20 | 30 | 4 | 26 | 11,557 | 11,468 | 430 |
| 7 | 20 | 40 | 4 | 26 | 11,950 | 11,848 | 578 |
| 8 | 20 | 50 | 4 | 26 | 10,583 | 11,798 | 108,000 |
| 9 | 20 | 75 | 4 | 26 | 10,165 | 10,522 | 108,000 |

## B.3 Instance 3

**Table B.3:** Increasing Possible Controller Locations: Instance 3

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 20 | 5 | 4 | 26 | 11,580 | 12,688 | $< 1$ |
| 2 | 20 | 10 | 4 | 26 | 10,359 | 10,670 | 2 |
| 3 | 20 | 15 | 4 | 26 | 9,264 | 10,711 | 532 |
| 4 | 20 | 20 | 4 | 26 | 9,320 | 10,259 | 8,374 |
| 5 | 20 | 25 | 4 | 26 | 9,523 | 10,333 | 23,723 |
| 6 | 20 | 30 | 4 | 26 | 10,485 | 10,562 | 429 |
| 7 | 20 | 40 | 4 | 26 | 11,072 | 11,623 | 108,000 |
| 8 | 20 | 50 | 5 | 30 | 12,273 | 12,844 | 108,000 |
| 9 | 20 | 75 | 4 | 26 | 10,757 | 11,689 | 108,000 |

## B.4 Instance 4

**Table B.4:** Increasing Possible Controller Locations: Instance 4

| | | | | | | CPLEX | |
|---|---|---|---|---|---|---|---|
| Problem | $\|S\|$ | $\|P\|$ | $\|P'\|$ | $\|L'\|$ | Packets | Cost($) | CPU(s) |
| 1 | 20 | 5 | 5 | 30 | 13,297 | 14,471 | $< 1$ |
| 2 | 20 | 10 | 4 | 26 | 9,815 | 10,400 | 9 |
| 3 | 20 | 15 | 3 | 23 | 8,877 | 8,999 | 5 |
| 4 | 20 | 20 | 5 | 30 | 12,011 | 13,203 | 77,457 |
| 5 | 20 | 25 | 4 | 26 | 11,348 | 11,800 | 1,468 |
| 6 | 20 | 30 | 4 | 26 | 10,110 | 10,321 | 168 |
| 7 | 20 | 40 | 4 | 26 | 10,320 | 10,348 | 1,641 |
| 8 | 20 | 50 | 4 | 26 | 10,355 | 10,189 | 1,919 |
| 9 | 20 | 75 | 5 | 30 | 12,033 | 12,675 | 108,000 |

# Appendix C

# Controller Placement Program Helper

```
$ python cli.py --help
Usage: cli.py [options]

Copyright 2014 Afrim Sallahi.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -c CONFSECTION, --confsection=CONFSECTION
                        The configuration section of where the input
                        parameters should be loaded.
  -a EBULK, --ebulk=EBULK
                        Configuration of solution files to do the
                           planning, eg
                        [10 20 30],[5 10]
  -b BULK, --bulk=BULK  Bulk generate/plot. switches then placements
     , eg [10
                        20 30],[5 10]
  -e, --expansion       Perform the expansion model. [default: False
     ]
  -i FILE, --in=FILE    The input directory [default: ../input]
  -o FILE, --out=FILE   The output directory [default: ../output]
  -x, --cplex           Generate the CPLEX files [default: False]
  -t, --script          Generate the SCRIPT files [default: False]
  -p, --plot            Plot the result file [default: False]
  -s FILE, --settings=FILE
                        The overall input file [default: ./config.
                           txt]
```