Understanding the Requirements for Successfully using Transfer Learning in Genetic Algorithms

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
with a
Major in Computer Science
in the
College of Graduate Studies at
University of Idaho
by

Tami Alghamdi

Approved by:
Major Professor: Terence Soule, Ph.D.
Committee Members: Jim Alves-Foss, Ph.D.; Jia Song, Ph.D.; Marshall (Xiaogang) Ma, Ph.D
Department Administrator: Terence Soule, Ph.D.

May 2022

Abstract

This dissertation is about understanding the requirements for successfully implementing Transfer Learning (TL) in the Genetic Algorithms (GA). TL is the procedure of transferring previous knowledge from an old problem, called the source problem (S) to another problem called the target problem (T). We have performed this study by implementing the process of the TL by employing the Genetic Algorithm (GA) as the model solver. GA is a type of Evolutionary Computation (EC) inspired by biological evolution theory that using biological evolution strategies by mimicking inheriting characteristics over many generations. TL has some limitations, for example, negative transfer. This situation halts the performance of solving the target problem. Also, during our study, we found out transferring the whole final source population to the target problem is not always a beneficial strategy for solving hard or non-related problems. Our study focuses on understanding the behavior of the transferred population and how to make them more beneficial to the target solver and the GA.

In this dissertation, we experimented with and evaluated several strategies for transferring knowledge including the Estimation of Distribution Algorithm (ED). We proposed an algorithm that samples the transferred population, and we evaluated our algorithm against other strategies of TL. We experimented and analyzed the effect of the content of the transferred population on the performance of the target solver. We experimented with transferring partial knowledge from the source problem to the target problem. We also experimented with sampling and transferring knowledge from multiple source problems to the target problem. The results of our studies show how TL can improve the performance of the GA in terms of the number of generations, time, effort the GA solver took to find the optimal solution. Also, analyzed factors that affect the GA performance and how to sample transferred population in terms of providing the GA with needed knowledge from the previous problem.

Acknowledgments

Alhamdulillah for everything. I would like to thank Allah for giving me success during my study. Without Allah's blessings, help and success, this dissertation would not be complete.

I am so grateful to thank and appreciate my father and mother for raising me and keep supporting me.

I would like to thank professor, Robert B Heckendorn, for his support, encouragement, correction, explanation, and patience with me. During all of my years in UoI, he was available for my questions and concerns.

I would like the thank my supervisor professor, Terence Soule, his advice was so valuable to me. I am proud of being one of professor Terence's students.

I would like to thank Professor, Jim Alves-Foss. For his advice, teaching me the fundamentals of research class, and how to write and conduct research.

I would like the thank Professor, Jia Song, for her advice and encouragement during my study.

I would like the thank Professor, Marshall (Xiaogang) Ma, for his advice, teaching me the Data Science class, and how to analyze, understand, and explain the data.

I would like to thank all my instructors for their hard work and dedication in providing me with a comprehensive and valuable education.

I would like to thank all the staff of the College of Graduate Studies for their help and support throughout my study at UoI.

I would like to thank many other friends who made my life at UoI very enjoyable.

I would like to thank and acknowledge my government Kingdom of Saudi Arabia, Saudi Arabia Culture Mission (SACM), and Al Baha University for their support and funding my studies in UoI and finishing this dissertation.

Finally, I am proud to be student in the Department of Computer Science at University of Idaho.

To my dear parents .....

Contents

List of Figures

List of Tables

List of Abbreviations

TL Transfer Learning

ML Machine Learning

EC Evolutionary Computation

GA Genetic Algorithm

S Source problem

T Target problem

$\mathcal{X}$ Feature space

$P(X)$ Marginal probability

$\mathcal{T}$ Task

$f(\cdot)$ Predictive function

$\mathcal{Y}$ Label space

X Feature vectors

x_i Feature vector

ED Estimation of Distribution

S_T Source Task

T_T Target Task

Chapter 1

Introduction

Humans tend to share knowledge of their experiences in order to facilitate their life. These experiences of what they have faced will pave the road and ease the process of solving new and hard life difficulties. Valuable knowledge will be passed from someone who had been in a situation and building on this knowledge will allow us to speed the process of finding new solutions. Life problems are not always the same but are often related to each other to each other. The famous adage "start from where others have stopped, and the lesson is in the endings" would summarize my thought of starting my dissertation.

In a real life we observe sharing the knowledge a lot. For example, if someone who used to play baseball, he can more easily learn how to play cricket than someone who plays soccer. Sharing the knowledge regarding related problems would make new problems easier and simpler than if there is no knowledge available. Sharing the knowledge will reduce the time to understand the problem and help with starting to find the solution from scratch. At least by sharing the knowledge we can understand what the problem is. Sharing the knowledge will give us a direction of how related problems were solved and how others thought about the solution. We can find how much old problems were related and from there we can start our process of solving the facing problem.

In a Computer Science field, knowledge sharing is well known as Transfer Learning (TL). In 2005, the Broad Agency Announcement of the Defense Advanced Research Projects Agency's Information Processing Technology Office gave a new mission to transfer learning: "the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. In this definition, transfer learning aims to extract the knowledge from one or more source tasks and then apply the knowledge to a target task. Traditional machine learning techniques only try to learn each task from scratch, while transfer learning techniques try to transfer the knowledge

from other tasks and/or problems to a target task when the latter has few high-quality training data."

Machine Learning (ML) algorithms usually require a huge amount of data to be trained, learn the task and start predicting the future. For example, if we want to train a model to classify images " cats and dogs ", we must have huge number of images cats and dogs'. Also, these images must be labeled accurately as cat or dog.

Machine learning train data must be collected from the same problems. We cannot train the model on images of cats and dogs and test it against images of apples and peaches unless we use TL. The model will not work in this case. If we want to train the model on a different image, we must find huge amount of accurate data first.

Traditional ML algorithms usually start their task from scratch to learn the task. We must prepare the data, train the model, and test the model in order to make sure it can predict accurate results. This process must happen each time when ever we want to use the model. This process wastes time and requires a lot of effort. Machine learning techniques require labeled and unlabeled data distribution to be the same. To compare, TL accepts the problems, tasks, and data distributions to be from different sources across the training and testing process.

Transfer Learning algorithm is a type of machine learning algorithm but does not require a huge amount of data. TL does not require the trained data to be from same problem. Transfer Learning can process its task on data that was collected from a similar or related problem not the same problem. Also, TL will not start from scratch to learn the task. TL overcomes the above preparation steps, and that results in the time performance, efficiency and the smooth performance of the ML algorithms.

We propose this dissertation to study how Transfer Learning (TL) will have great effects and impacts in terms of improving the accuracy, efficiency, and fast solution production of Evolutionary Algorithms (EA), especially Genetic Algorithms (GA).

1.1 Transfer Learning

Transfer learning (TL) is a type of Machine Learning. It is the process of transferring the gained knowledge from one problem to a similar problem. This can expedite training perhaps make for more robust solutions. Nowadays machine learning algorithms are designed to learn and train to solve one specific problem. When the same algorithm is facing similar problem but the algorithm is different in its feature space or different marginal probability the algorithm

must start from scratch to solve the problem. Transfer learning is the process of overcoming this problem by using gained knowledge from old or previous problem and placing them in the new problem. This dissertation provides an overview of transfer learning, discusses types and approaches of transfer learning, and explains negative transfer and how to apply transfer in Genetic Algorithm.

The idea behind transfer learning is to mimic human innate behavior. When a human is facing a new problem, the human will think of what is the closest problem they have solved to the new problem and rely on that knowledge to help solve the new problem. The old and new problems may not be the same, but they may share some features or aspects that can be leveraged. For example, if a person speaks two languages it is easy for him or her to learn a new one if the languages are similar, or a player who plays baseball can learn to play cricket more easily than someone who plays soccer. [69]

Figure 1.1, represents the process of TL. The source problem find a solution to the S problem. The TL transfer the knowledge of the source final solution to the target problem. The target problem uses the transferred knowledge as starting point to find a solution to the T problem.

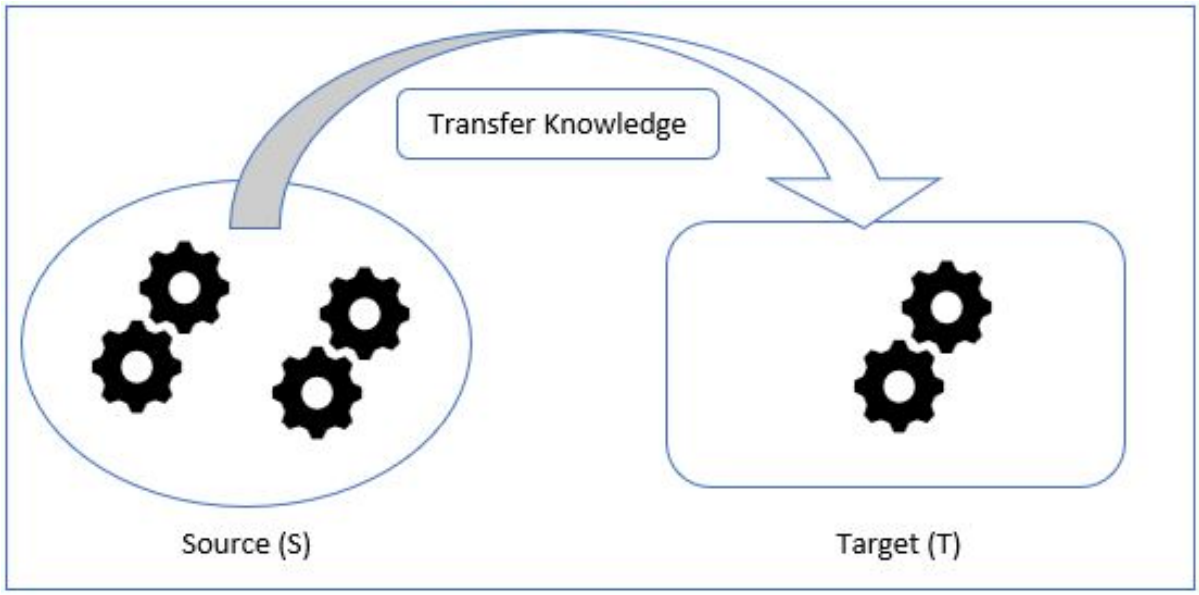

Figure 1.1: This diagram represents the process of TL. The final solution of the source problem transfers to the target problem.

1.2 Evolutionary Computation

Evolutionary Computation (EC) is a field of artificial intelligence inspired by biological evolution that used to find the global minima or maxima of a function and they are folk of algorithms. They are based on trial and error to solve problems type, where they generate random potential solutions to solve a problem. The initial pool of potential solutions is generated and updated many times based on evolutionary computation operations (mutation and cross over). Each new update removes less potential solutions or individuals, and adds new solutions to the initial pool. As the result of the removal and addition of solutions to the initial potential solution pools, those new solutions or individuals will inspire to be good solutions. EC is used in many fields to produce experimental procedures to practice and study how robust the solutions are. It is important and we believe it can reach the aims or dreams of Artificial Intelligence where machines can learn and be experts in their fields. [23]

1.3 Genetic Algorithm

Genetic Algorithm (GA) is a type of evolutionary computation, also inspired by biological evolution theory. GA is mimics biological evolution strategies of characteristics inherited over many generations. An evolutionary algorithm contains the following steps: initialization, evaluation, reproduction (crossover and mutation), and termination. For example, placing the individual who has fit fitness will live and reproduce, in contrast to other individuals who have low fitness who will die. When an individual or solution is created for first time, it will assign a fitness value based on a fitness function and its characteristic and will be placed in the selection pool. **Fitness function** is type of loss function used to select or determine how a particular individual or potential solution is close to winning the objective aim. Each individual will go through process if this individual has high fitness value will survive and reproduce again, otherwise it will be removed, this process is called natural selection. **Natural Selection** is considered to be the key element of evolution. It is the contrasting survival and reproduction of many potential solutions from the pool due to differences in their fitness value. The algorithm's steps and natural selection would interact together in order to implement the process of evolutionary algorithms. We believe evolutionary algorithms are a natural framework for exploiting transfer learning [21, 33, 49].

1.4 Motivation, Objectives, and Contributions

1.4.1 Motivation

Exploiting knowledge from previous experiences can reduce mistakes, effort, and time to achieve the solution of a novel, but related, problem. From a known problem and solution, we can aim for other harder problems and use the previous knowledge and other related knowledge to achieve solutions to the harder problems. Our study shows how to explore, mine, and collect well-trained data from different sources. All we need to make our machine fast and keep pace with massive technology development is well-trained data. This data can be found and prepared by using our study and TL.

Generally, Transfer Learning is a machine learning technique that uses pre-trained data in other problems. TL makes the process of solving new problems smoother and faster. Also, the solutions can be more robust, accurate, and reliable.

We have studied and demonstrated in situations where we do not have enough pre-trained data TL can play a major role. Problems with high computation resources can be solved by using TL. It can overcome the over-fitting data problem. Well-trained data availability is a problem and we may not have it available when we need it. The high cost and the standards of the job of labeling data take a lot of time and huge effort. This study showed TL can overcome these technical problems and others.

Transfer Learning can also solve the cold start phenomena. We do not have to waste time, effort, and start from the early beginning to solve a problem. The effort that may be spent to find the solution to a new problem can be minimized. Since we have a huge amount of data repositories available around us, we must get the benefit and use of these data repositories.

The scope of this research is to understand the proper method of how knowledge is transferred between problems. In this research, Genetic Algorithm (GA) is implemented as the solver of the problems. The problem's solutions will be studied more and prepared to be transferred as an initial population to the GA solver of the target problem. The transferred population is used as a help to find the solution and reduce the effort of the GA solver.

1.4.2 Objectives

The purpose of this research is to study how efficiently knowledge of a previous problem can be transferred to other problems. My study examines in detail about how previous knowledge can be found, collected, organized, prepared, improved, measured, and transferred. Also, we

evaluate our proposed studies to improve the performance of finding new solutions of the Genetic Algorithm. This study was implemented using the JAVA programming language. The following objectives are particularly covered by this study:

Objective 1. Modeling the Transfer Learning (TL) process by implementing the Genetic Algorithm (GA) as the model's solver and transfer the knowledge from the source problem to the target problem. The model allowed us to study the following tasks:

Task 1.1 : Compare five different strategies of transferred population. These strategies include the Estimation of Distribution (ED) concept.

Task 1.2 : Understand the role of diversity in the GA performance.

Deliverable : A published paper presented at the IEEE Congress on Evolutionary Computation conference (CEC-2021). [2]. Chapter 4.

Objective 2. Propose an algorithm for sampling the transferred population.

Task 2.1 : Study which part of the S data has the most information about solving the problem.

Task 2.2 : Does transferring these parts of the S population help find the T problem's solution easily?

Deliverable : A published paper presented at the 2021 World Congress in Computer Science, Computer Engineering, Applied Computing conference (CSCE-2021). Chapter 5.

Objective 3. Study how to quantify the right combination of the transferred population. This study will allow me to answer the following tasks:

Task 3.1 : Study the right quantities of each combination of the transferred population (old knowledge and population diversity).

Task 3.2 : Investigate what the other factors are that may affect this combination.

Deliverable : A published paper presented at the Computational Science Computational Intelligence conference (CSCI-2021). Chapter 6.

Objective 4. Analyze the effect on the content of the transferred population. We will perform the following tasks:

Task 4.1 : Prove that the shared information saved in the population after solving the source problem helps solve the target problem.

Task 4.2 : Investigate what are the important parts of the information from the previous populations needed to solve the target problem.

Task 4.3 : Study how critical is the relationship between the source S and the target T problems in the TL environment.

Deliverable : A research paper. Chapter 7.

Objective 5. Transferring partial knowledge from the source problem (S) to the target problem (T). This study will allow me to understand the following tasks:

Task 5.1 : Answer a concern of, does transferring partial knowledge from the source problem to the target problem help the target solver find the solution easily or not.

Task 5.2 : Study the influence of transferring partial knowledge of the source problem information to the target problem.

1. Deliverable: A research paper, submitted to the ICECA 2022: 16. International Conference on Evolutionary Computation and Applications. Chapter 8.

Objective 6. Provide a study of transferring knowledge from multiple different source problems ($S_1, S_2, S_3, ...$) to the target problem (T). This study will allow me to understand the following tasks:

Task 6.1 : Provide a case study of transferring knowledge from different source problems.

Task 6.2 : The possibility of transferring knowledge from multiple different source problems to a target problem.

Task 6.3 : Provide results of this study.

Deliverable : A research paper, submitted to the ICSEC 2022: 16. International Conference on Swarm and Evolutionary Computation. Chapter 9.

1.4.3 Contributions

Most of the previous works focused on transferring the final solution of the S problem to the target solver. Other studies improved the sampling process of the transferred population by reproducing the best individual of the source final solution a number of times. Other studies implemented the TL in other fields of ML, for example, Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Natural Language Processing (NLP), Computer-Human Interaction (CHI), and other fields of computer. The proposed research focused on the performance of the GA and how to transfer the needed knowledge.

This dissertation study and concentrate on understanding the requirements for successfully using Transfer Learning (TL) in Genetic algorithms (GA). Under these situations we must pay much attention to these requirements. What strategies help the TL designers to sample the transferred population that will improve the GA performance? How to find the best knowledge from previous solutions? How to sample the transferred population accurately and efficiently?

This dissertation analyzed what factors must be available in the transferred population. This study discovered the elements that must be obtainable by GA to find the solution to the T problem more easily and efficiently. This study would make the performance of the GA more reliable, especially when transferring the knowledge based on this study.

This study can be implemented as a tool for finding and sampling accurate subsolutions to be transferred to the GA solver. This study can be implemented in medical fields, robots movements, and other decision making applications. Wherever there is a use of GA this study can be used to improve the performance.

1.5 Dissertation Impact

Machine Learning and Transfer Learning designers can benefit from our work by leveraging what this research shows about solving related problems. They do not have to start from scratch in order to use transfer learning to solve a problem. The algorithm designers can use our study to improve other algorithms, especially Genetic Algorithms, by including a transfer learning step. As repository data sets are available around the internet, they can get the benefit of existing data and solutions to leverage the process of getting newer or more robust solutions.

We believe this study can be implemented wherever there is a use of GA. The medical and industrial fields are where we think the most benefit will be found. For example, If a cancer medical research center is using the GA to detect one type of cancer tumor, they can apply the results of our study to help train systems to detect other cancer tumor types for effeciently. As the system already has knowledge of detecting one type of cancer, they can use this study to help train their system to detect another type of tumor. Similarly, the human immune system is different from person to person, medicine should be tailored from patient to patient. We can use this study as a tool that transfers knowledge and finds the best prescription based on the patient's fitness.

Robots and transportation systems can be improved by implementing our study in their system. Self-driving cars make good decisions sometimes but are not always right. For example,

self-driving cars navigate through uncrowded roads. By implementing the study, we can transfer the knowledge between cars, which will allow cars to more quickly learn to make better decisions in novel real-world scenarios.

1.6 Dissertation Overview

This dissertation is organized as follows. Chapter 2 provides background and common terminologies of TL that we used in this dissertation. Chapter 3 provides literature review of old studies of TL. Chapter 4 provides a study of five different strategies of transfer learning including Estimated of Distribution algorithms (ED). Chapter 5 presents a proposed algorithm of how best to sample the transferred population and evaluate the performance of our proposed algorithm with other strategies of TL. Chapter 6 talks about quantifying the right combination of knowledge and population diversity for transferred populations. Chapter 7 presents an analysis of the effect of the content of the transferred population from the source problem on the performance of finding the solution to the target problem. Chapter 8 discusses transferring partial knowledge from the source final solution to the target problem. Chapter 9 talks about sampling and transferring the transferred population from multiple source problems to the target problem. Chapter 10 concludes this dissertation and provides some directions for the future work.

Chapter 2

Background and Terminologies

2.1 Introduction

This chapter provides the background of Transfer Learning and reviews relevant terminology along with this dissertation. We define the Transfer Learning (TL) terminology and other concepts of TL and Machine Learning (ML), for example, source, target, task, and other. We talk about the TL questions that must be answered before the TL process begins. We explain the limitations of TL. Lastly, we demonstrate the types and approaches of TL.

In the beginning, TL deals with two domains: the source problem and the target problem. Each domain has its own task.

The **source domain** is where machine learning methods learn from scratch and it has the following symbol:D_s.

The **target domain** is the new similar problem, where knowledge will be placed, and it has the following symbol:D_t.

The **domain** has the following symbol $\mathcal{D}$, and it has two elements:

- Feature space $\mathcal{X}$.
- Marginal probability $P(X)$, where $\mathcal{X}$ is the sample data points.

in the domain $D = \{\mathcal{X}, P(X)\}$ as we mentioned above $\mathcal{X}$ is the feature space, and $P(X)$ is the marginal probability over feature space, $X = x_1, x_2, x_3, \ldots x_n \in \mathcal{X}$. $\mathcal{X}$ is representation document of all space, X sample documents used for training, and x_i term vector corresponding to some document.

A **task** $\mathcal{T}$ to be learned is defined as $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$, where $\mathcal{Y}$ is the label space, and $f(\cdot)$ is the predictive function. The task is not monitored and will learn from the training data.

The **training data** is a set of pairs $\{x_i, y_i\} : x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The predictive function, $f(\cdot)$, attempts to predict the value $y \in \mathcal{Y}$ given a value from x.

The **training inputs** consists of a list of k features. A single sample from the training inputs is a **feature vector** x_i which is a k dimensional vector of values for the features.

A **training set**, X, is a set of feature vectors $X = \{x_1, x_2, x_3, \ldots x_n\}$.

In a general view, the differences between domains can be known by different feature space or different marginal probability distributions. Each domain has its own tasks, and these tasks are different. For example, we will take classification as an example: y consists of all "True" and "False" values, and $f(\cdot)$ learn the relationship between domain X and y label space during the training process. [53]

Transfer Learning TL is a Machine Learning (ML) process that aims to improve the target domain's predictive function $f_t(\cdot)$, by applying the knowledge from the source domain D_s and the target domain D_t where the source domain D_s data and the target domain D_t data are not the same ($D_s \neq \mathrm{D}_t$), or the source task $\mathcal{T}_s$, and the target task $\mathcal{T}_t$ are not the same($\mathcal{T}_s \neq \mathrm{T}_t$).

2.2 Transfer Learning Questions

The following questions need to be answered in order to achieve the benefits of transfer learning:

1. When to transfer?
2. What to transfer?
3. How to transfer?

It does not matter if we are doing Transfer learning in Evolutionary Computation or other technology fields. The above questions must be answered.

In order to answer question one, source domain and target domain must be related to each other. In some cases when a source problem is different entirely from a target problem, transferring knowledge will not help and would make the target learning process slow and inefficient; this situation is called negative transfer. Most of the previous work considers questions two and three (what and how to transfer) and assumes by default there is a relation between source and target domain. Recently, researchers have proposed the idea of transfer learning that can happen even if there is no relation between source and target, like Ben Tan et. al. in their paper "Distant Domain Transfer Learning" [53, 65].

What to transfer is the second question and must be answered, after the problem solved in the source domain asks which part of the knowledge must be transferred to the target domain. Some of the knowledge is specific for source domain and there are not beneficial aims of transferring them to the target domain. Some of the knowledge is common between source domain and the target domain and these data or knowledge is targeted and must be transferred to the target domain [53].

How to transfer is the third question that must be answered. The problem is not yet solved by discovering the targeted knowledge from the source domain. After knowing which part of the source domain's knowledge is selected to be transferred, a learning algorithm must be coded to transfer selected part of the knowledge to the target domain. [53]

2.3 Transfer Learning Limitations

The limitations of the TL can be categorized into the following two topics: negative transfer, and ineffective transfer.

2.3.1 Negative Transfer

Negative transfer is a situation where transferring knowledge would make target domain's process slow and inefficient. This situation happens when the source domain and target domain are not the same or when the tasks of each domain are not the same. Forcing knowledge to target domain will not be helpful and might create problems. People try to solve this issue by inviting new ideas like measuring how much data can be transferred to the target. These values are not always helpful and, in some research, did not perform well. These values are called k adaptive value where this k value tries to measure the direct relation between source and target [7, 43, 53].

2.3.2 Ineffective Transfer Learning

Ineffective transfer is a situation where the target problem can not be explored with a single property of transferred data. The transferred knowledge is underfitting the target task problem or needs. For example, in the graph transfer learning, the statistical property and geometric structure are two important factors that must be transferred together. If we transfer the geometric structure alone to the target problem this transferred data is not enough to complete the target task [40].

2.4 Types of Transfer Learning

According to previous researches and studies, TL can be categorized based on two criteria: data availability and feature space similarities.

Based on the data available as labeled data in the source and target problem, TL can be categorized into the following three categories [53]:

- Inductive transfer
- Transductive transfer
- Unsupervised transfer

2.4.1 Data availabilities

Based on source domain, source tasks, target domain, and target task, these types have some common and different features.

2.4.1.1 Inductive Transfer

Inductive transfer learning is a type of transfer learning where source domain and target domain are the same or not (it does not matter), but source task and target task are not the same. Inductive type has two situations regarding the sources' labeled data:

- Source domain has a lot of labeled data. In this situation inductive transfer learning aims to transfer knowledge to increase the performance in the target domain and its predictive function.
- Source domain labeled data is not available. In this situation source domain labeled data cannot be used and if forced on the target domain to used source domain data, we would face negative transfer situation.

In both cases inductive transfer learning tries to increase performance and efficiency of the target domain. For example, we are trying to classify images either an image has a car or not. Let's say we have 10,000 image we have 1000 image has car, and 1000 do not have car. The rest 8000 images we do not know for sure if they have car or no. Inductive transfer works by looking at the labeled 2000 images and building a classifier for it [10, 53].

2.4.1.2 Transductive Transfer

Transductive transfer learning is a type of Transfer Learning where source domain and target domain are different, but source tasks and target tasks are the same. In addition, some of unlabeled target data must be available during the training process. This type has two situations regarding feature space:

- Different feature space between source and target domain $X_s \neq X_t$
- Same feature space between source and target domain, but different marginal probability distribution of the input data. [53] [14]

For example, we will use the example of car classification from above. Transdictive transfer learning works by looking to the 8000 images trying to find information about the label space.

2.4.1.3 Unsupervised Transfer

Unsupervised transfer learning is a type of transfer learning where source domain and target domain are the same, and the target task is different than the source task, but they are related to each other. This type of transfer learning does not require available labeled data in both domains, also this type focuses more on target domain by solving unsupervised learning task (for example, (clustering) [53] [11].

2.4.2 Data Availability Transfer Learning approaches

Transfer learning approaches can always be used to answer the question "what to transfer". These approaches are the following:

- Instance-transfer
- Feature-representation-transfer
- Parameter transfer
- Relational-knowledge-transfer

2.4.2.1 Instance transfer

Instance transfer learning is an approach of transfer learning that can be used in inductive and transductive types of Transfer Learning. In this approach, the source domain data can not be reused directly in the target domain. Some portion of the sources' data with labeled data

can be used in the target domain, but the other portion of the data is not helpful and may cause a problem. [53] [9]

2.4.2.2 Feature Representation

Feature representation transfer is an approach of Transfer Learning that can be used in all types of transfer learning (inductive, transductive, and supervised). This approach aims to find good feature representation for target domain. This representation would reduce the variance and error rate between source domain and target domain. [53] [5]

2.4.2.3 Parameter Transfer

Parameter transfer is an approach of Transfer Learning that can be used in inductive transfer learning type. Finding some parameters or prior distribution of the source domain and knowledge can be encoded to transfer to the target domain. By default, one must assume that source task and target task share some, not all, parameters or some prior distributions of the data. [53] [24]

2.4.2.4 Relational Knowledge

Relational knowledge transfer is an approach of Transfer Learning that can be used in inductive transfer learning type. Also, this approach deals with relational domain. In fields like network or social network, data are non-independent and identically distributed (i.i.d) as traditionally assumed, and the relation among them can be represented by multiple relations. This approach transfers the relationship between data in the source domain to target domain. [53] [44]

Table 2.1 shows which approach is suitable with different types of transfer learning:

Table 2.1: Transfer Learning Types and Approaches of Data Availability

Types / Approches	Inductive	Transductive	Unsupervised
Instance transfer	✓	✓	
Feature representation transfer	✓	✓	✓
Parameter transfer	✓		
Relational knowledge transfer	✓		

2.4.3 Feature Space Similarities

Weiss and et al. [74] categorized Transfer Learning based on the feature space similarities of the source domain and the target domain ($X_s = X_t$). The similarities of domains can be

detected if the source domain and the target domain are subdomains of a general domain. For example, musicians can learn different musical instruments easily because they are subdomains of a common general domain which is music. Based on these criteria, the authors divide TL into the following two types:

- Homogeneous Transfer Learning
- Heterogeneous Transfer Learning

2.4.3.1 Homogeneous Transfer

Homogeneous transfer learning is the case where the source domain feature space is the same as the target domain feature spaces ($X_s = X_t$), and the source domain label data is the same as the target domain label data ($Y_x = Y_t$). The source and target domain must be related, or they are subdomains of a common domain.

In the case of Homogeneous transfer learning, the task of the solution will be one of the following tasks:

1. Solving the marginal distribution variation between the source domain and the target domain $P(X_s) \neq P(X_t)$.
2. Solving the conditional distribution variation between the source domain and the target domain $((P(Y_s|X_s) \neq P(Y_t|X_t))$.
3. Solving the marginal and conditional distribution of the source and target domains.

2.4.3.2 Heterogeneous Transfer Learning

Heterogeneous Transfer Learning is the case where the source domain feature space is not the same as the target domain feature space ($X_s \neq X_t$), and the source domain label data is not the same as the target label data ($Y_x \neq Y_t$). Both domains are derived from different domains.

In the case of Heterogeneous transfer learning, the task will be to degrade the problem to Homogeneous transfer learning, and from there, the task will be to solve the marginal or conditional distribution of the source and target domains.

2.4.4 Feature Space Similarities Transfer learning approaches

The authors of the survey mentioned five approaches to solve Transfer learning: instance-based, feature-based, parameter-based, Relational-based, and Hybrid-based. The first four approaches

are the same as the data availability Transfer learning approaches Section 2.4.2. However, the exception is the feature-based approach which uses two different types: asymmetric and symmetric. Also, they introduced a hybrid-based approach.

1. The asymmetric type is a case where the conditional probability is different between the source and the target domains $P(Y_s|X_s) \neq P(Y_t|X_t)$

2. The symmetric type is a case where the marginal distribution is the same between the source and the target domains $P(X_s) = P(X_t)$. The goal is to find common latent factors between both domains.

2.4.4.1 Hybrid-Based

The hybrid-based approach is a mixed approach between the instance-based and parameter-based approach. The goal of this approach is to solve the marginal distribution first and solve the conditional distribution second between the source and the target domains. This type of approach is applied to solve Homogeneous transfer learning types.

The following table summarizes the approaches of solving the second type of Transfer Learning feature space similarities type.

Table 2.2: Transfer Learning Types and Approaches of Feature Space Similarities

Types / Approches	Homogeneous TL	Heterogeneous TL
Instance-based transfer	✓	
Feature-based transfer	✓	✓
Parameter-based transfer	✓	
Relational-based transfer	✓	
Hybrid-based transfer	✓	

Learning categorizations.jpg

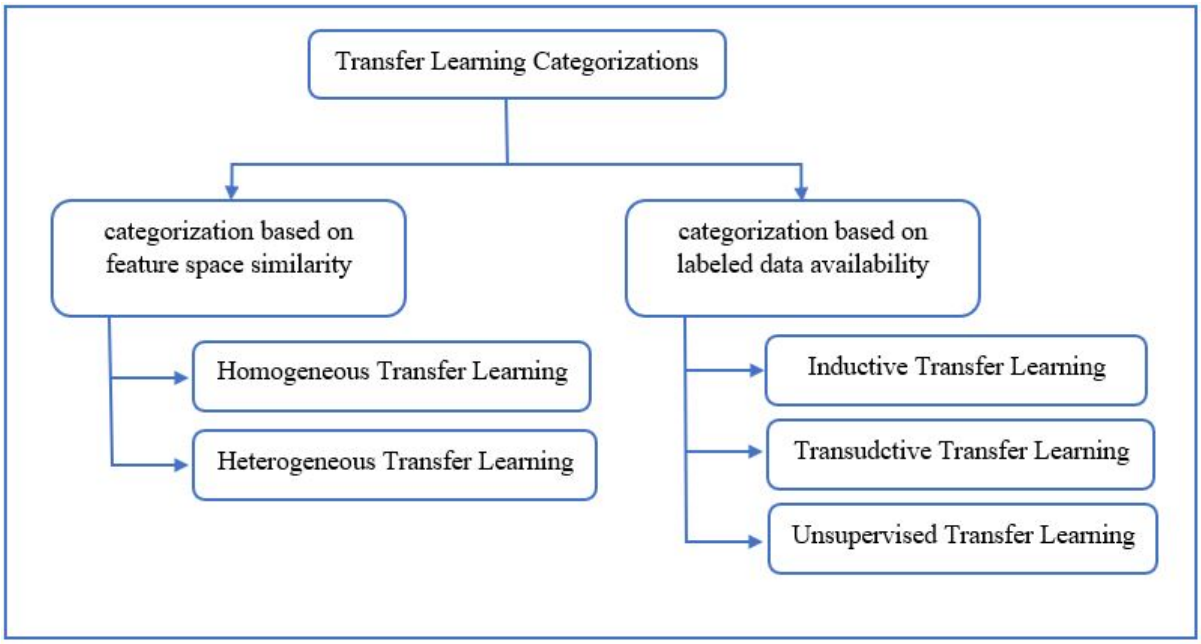

Figure 2.1: This diagram categorizes Transfer Learning into two categories based on the availability of the labeled data, and feature space similarities.

Chapter 3

Literature Review

The goal of this chapter is to provide some of the old studies that are related to our topic. These studies have the most related ideas and approaches that we followed to accomplish this dissertation. Some of these studies are books, dissertations, and published papers.

Fangtao et al [37] implemented TL with extracting sentiment and topic lexicons. Their study aimed to employ information from the source domain to the target domain. They proposed a two-stage TL method:

1. seed sentiment: a bridge between the source and the target domain that identifies mutual sentiment words in the target domain. Then creates topics in the target domain by finding the relation between sentiment words and topics from the source domain.

2. Relational Adaptive bootstraPping (RAP): a proposed method that extends the target domain seeds.

The authors claimed that their proposed algorithm can utilize information from the source domain and exploit the relationships between topic and sentiment words to the target domain. They evaluated their work using a reviewer dataset that contains 500 movies and 601 product reviewers. The results showed that the proposed method was effective compared to other methods.

[13] Daumé III, Hal proposed a Transfer Learning method called Feature Augmentation Method (FAM). This method based on the target domain must have some labeled data and, copying the feature spaces of the source and target domains a fixed number of times. In Daume's work, FAM copied each data of the source and target domains three times. For the transforming task, they transform the source and target augmented data to the learning method. His work tested against several datasets like ACE_NER, CoNLL-NE, PubMed-POS, and other datasets. the results showed FAM performed great.

[52] Pan, Sinno and et al proposed a Transfer learning method called Transfer Component Analysis (TCA). This method based on the feature extraction approach also required the source and target domains to have data. The target domain did not require to have labeled data (Y_t). Basically, TCA attempt to discovered common latent factors or components between the source and target domains and transfer these factors. For example, it tried to discover the difference of the data distribution between the source and the target domains. They tested their work by learning 1-D from 2-D data, they used Wi-Fi dataset. The results showed a great work of TCA reducing the distribution distances between the source and target domains data.

[24] Gao and et al have proposed a graph-based method of Transfer Learning. This method is based on assigning an optimal weight of the source domains. The optimal weights are calculated periodically based on the source domain data. They transferred the optimal weight. They claimed this method reduces the risk if the weights are set optimally. The authors experimented with this method against four real datasets like, Email spam filtering, document classification, and others, also they tested it against one synthetic dataset. The results showed that the knowledge of the source domains can be identified by calculating and transferring the optimal weights.

Mihalkova et al [44] implemented TL with Markov Logic Networks (MLN). They proposed a system called TAMAR that utilized information between two different domains. The proposed system performed the following two subtasks:

1. Mapping: establish cluster mapping between the source and the target.

2. Re-weighting: learn from the source domain how to reweight the previous clusters and apply them in the target domain.

The authors evaluated their system using three different datasets. The results showed TAMAR system learns faster and can be used to improve the accuracy of other models.

Pan et al [50] implemented TL with Dimensionality Reduction. They proposed a method called Maximum Mean Discrepancy Embedding (MODEL) that minimizes the distance between the feature distribution of the source and the target data. The proposed method performed the following two steps:

1. Discover low dimensional of common features for the source and the target domain.

2. Train the model by using the discovered data from the first step, and traditional machine learning algorithm.

They evaluated their work using two different experiments: the first one used Wifi data, and the second experiment used binary text classification data. The results showed that the MMDE method improves the performance of ML algorithms compared to standard algorithms.

Long et al [41] presented transfer learning with Joint Adaptation Networks (JAN). JAN exploited the adaptation ability of Deep Neural Networks (DNN). They began by using DNN to operate the source and target data and then align the operation of the joint distribution. They suggested the Joint Mean Maximum Discrepancy (JMMD) method to calculate the distribution joint of the source and target data. They performed their study using standard datasets like office-31, imageCLEF-DA, ALEXnet, and ResNet. The authors claimed that JAN is able to correct the joint distribution shift of each layer in the networks. The results of the experiments showed JAN performed great and benefit from Transfer Learning.

[80] proposed a Heterogeneous Transfer Learning Image Classification(HTLIC) method. The main contribution of their method is to enhance the performance of the target domain by using unlabeled data of the source domain. HTLIC method is operated using the following three steps:

- Build a connection between source and target data.
- Use Collective Matric Factorization (CFM) to learn relationships between source and target.
- Construct new relationships between the source and the target domain.

The Caltech-256 dataset was used to run the experiment. The results showed HTLIC outperformed other baseline methods like Orig, PCA, and Tag. The authors claimed that HTLIC can learn by using data like textual and tagged information from social websites like Flicker.

[51] have studied Transfer Learning in the sentiment classification field. The authors have proposed a Spectral Feature Alignment (SFA) algorithm to minimize the gap between the source and the target domains. the SFA algorithm solves the differences of the marginal distribution between the source and target domains by learning new feature representations of both domains. SFA constructs a bipartite graph by creating two domains: domain-specific words and domain-independent words, to find out the co-occurrence relationships. That is, if two or more words of domain-specific linked with two or more words of domain-independent words these words aligned together. The authors have evaluated SFA using products reviews data from Amazon,

the other dataset they constructed from Amazon and other websites. The results showed how SFA benefited from Transfer learning and classified sentiment words accurately.

[27] studied Transfer Learning in the object recognition field. The author's proposed two methods: the first one is a kernel-based method called the Geodesic Flow Kernel (GFK). GFK method operates by gaining the advantages of the low dimensional structures of the data. The second method is a Rank of Domain (ROD) which assumed the target domain does not have labeled data. ROD is a method that sorted the source data based on which data is appropriate for the target domain. They tested their work using three different datasets. The results showed GFK outperformed other object recognition methods.

Brain-computer interfacing (BCI) is a field that allows a patient with a disability to communicate with others or closed objects. Affective BCI requires a massive amount of collaborations between other systems. For example, human brain tasks like word generation or mental rotation required Electronic Ephalo Graphy (EEG) and functional Transcranial Doppler ultrasound (fTCD) systems to record brain electrical signals and cerebral blood velocity. [34] Khalaf et al proposed Transfer Learning to reduce the amount of collaborations. Also, they used the mutual information to select the best three datasets used by other patients. They tested their model using two human activities (word generating and mental rotation). The authors constructed three experiments: the first one is to distinguish between word generation and mental rotation skills; the other two experiments were to discriminate between the two selected skills against the baseline. The results showed the performance of the model was faster, more accurate, and allowed fewer communications.

[61] Sun et al, performed TL with mutual information. They proposed an infoGraph method that maximized the mutual information between graph-level representations and different scale substructure representation like(object, node, edge). They aimed to mark data that are shared between graph-level representation and different substructure scales. They ran the experiment using six well-known benchmark datasets like MUTAG, PTC, and others. The results showed the infoGraph method is superior to state of the art methods. Also, the authors improved the infoGraph method for semi-supervised cases and named it infoGraph. They used the same idea of maximizing the mutual information between known supervised methods and data that run using the infoGraph method. The results showed infoGraph is competitive with other methods.

[71] Wang et al, implemented a classification model that used Transfer Learning strategies. Conventional Transfer Learning strategies aim to learn knowledge about or representations of

the source domain and use it in the target domain task. The authors tried to make the learning process more generalized. Instead of taking data belonging to one source domain, they learned from many different source domains. They achieved this goal by doing the following steps:

- They learned the representation of domain-independency by minimizing the mutual information between data of the source and the target domains.
- They learned the discrimination of the data by maximizing the marginal of each data point.

The authors ran the model using four different benchmark datasets like Office-31, Email spam, ImageCLEF-DA, and Amazon. The results showed the model is efficient enough over the benchmark datasets.

Torkkola and Campbell [68], used Maximum Mutual Information (MMI) in feature reduction or transformation. They proved mutual information overcame some limitations of other methods like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). They claimed mutual information can be used as a general standard and it can be used as higher-order statistical. The authors created two models: pattern recognition and data analysis. They tested their models using a UCI machine learning repository dataset. The results proved that mutual information is a useful tool for transfer data to low dimension data.

Pablo et al. [22] proposed a feature selection method built on mutual information. They named the proposed method Normalized Mutual Information Feature Selection (NMIFS). They aimed to measure the redundancy, consistency, and relevance of the features using Mutual Information. They ran an experiment using four datasets. The results showed NMIFS method is fast and efficient, but it is limited to one feature. The authors upgraded the previous method and utilized the genetic algorithm (GA) to improve the performance to select group of the related features. They named the updated method genetic algorithm guided by mutual information for feature selection (GAMIFS). They ran the updated version of the proposed method and the results showed it can find a single feature also can find groups of features that are related.

[25] proposed an algorithm that implemented Transfer Learning in sentiment words analysis. Their proposed algorithm used deep learning and transfer learning to perform its analysis. The algorithm consisted of two steps:

- Use Stacked Denoising Autoencoder(SDA) to discover latent features space between the source and the target domain.

- Train the model using latent features data from the previous step with labeled source data.

The training process was done by using the Support Vector Machine (SVM). The model evaluated against Amazon products and a large industry dataset. The result showed the proposed algorithm outperformed other sentiment classification algorithms.

[55] authors implemented Transfer Learning in Bayesian optimization to avoid what is known as the " cold start" scenario. They used a probabilistic model (Thomson sampling) to obtain the optimal solution. The authors proposed to define the similarities by measuring divergence between the source and the target distributions. Then these distributions were added together to be used in the acquisition function or learned knowledge. The predictive Entropy Search was used to build learned knowledge. They evaluated their work using the LibSVM repository which consists of five binary datasets. The results showed how effective this model was compared to other current methods of Transfer Learning.

Qian Sun et al. [62] proposed two stages of reweighted algorithms. They aimed to calculate instances of different source domains and re-weighted the marginal and conditional probability of the source and the target domains. They named the proposed algorithm 2-Stage Weighting Framework for Multi-Source Domain Adaptation(2SW-MDA). Their work consisted of the following two main steps:

1. Reduce marginal probability: calculate and reweigh source domain instances using Maximum Mean Discrepancy (MMD).

2. Reduce conditional probability: calculate and reweigh instances of multiple source domains based on target domain smoothness assumption.

The authors evaluated their work using three data sets. The results showed this algorithm effective compared to other transfer learning methods like Domain Adaptation Machine (DAM), Locally Weighted Ensemble(LWE), and other methods.

Nagae et. al [48] proposed a method that used TL and GA to improve the performance of Convolutional Neural Networks (CNN). The method spontaneously finds and chooses the best layer of the CNN and transfers it to the target problem. The evaluation of this method was performed using ImageNet as the source problem and the CIFAR-100 data set as the target problem. The result showed the accuracy of using the proposed method higher than the

accuracy of using classic CNN. Even though they tried different types of selection methods, the accuracy results were higher than the classic CNN.

Jiang and Zhai [32] implemented TL with Natural Language Processing(NLP). They proposed a heuristic instance weights method. The authors claimed that their method can find the difference between the source and the target domains. The results showed that using information from the target domain will improve the proposed method's performance. They evaluated their method using three different tasks of NPL. The results showed that the performance of the proposed method overcomes supervised and semi-supervised learning methods.

Zadrozny [79] proposed a bias correction method. This method predicts the corrected distribution of unbiased data by subtracting the rejection sample from the selected sample. From the TL standpoint, this method can be used to select the transferred populations between the source and the target domains. The author concluded that there are two types of sample selection bias classifier:

1. local: that depends on the predictive function.

2. Global: that depends on the predictive function and marginal distribution.

The author used the KDD-98 dataset to evaluate the method. The results showed this method can be used to classify data.

Jiayuan at el [30] proposed the Kernel Mean Match(KMM) method. This method finds the resampling weight without known true distribution sampling. The authors claimed the KMM method overcomes the covariate shift sampling problem. Before the KMM recalculates the training data by using the kernel Hilbert space of the means of source and target problems, the KMM learns the ratio between the source and the target data. They used the UCI benchmarks datasets to evaluate the KMM method. The result showed that the proposed method performed well, and, in some cases, it outperforms the known distribution.

Koçer and Arslan [1] proposed a hybrid GA and TL algorithm. They evaluate their algorithm using neural networks. The neuron weights were updated using their algorithm. They created a solution pool that collects three individuals (the best, the middle, and the worst) fitness value of each iteration of the source task. Then they transfer the solution pool to the target problem. The authors evaluated their algorithm using the UCI dataset. The result showed the proposed algorithm performed well. Also, it has some advantages like ease of implementation, and it can be used in other domains. For example, it can be used to initial and update the

neuron networks weights.

Loey et al [39] implemented TL with classification. They proposed a TL mask face detection model. This model can be integrated with monitoring cameras around cities or closed places to observed people who are wearing masks for others. This model is based on deep transfer learning and classical machine learning algorithms. The proposed model has the following two tasks:

1. feature extraction, through deep transfer learning.

2. data training, though classical machine learning algorithms. For example, decision tree, SVM, and ensemble.

The authors evaluated the model using three different datasets. The results showed this model detected the face masks with high accuracy. They concluded that deep transfer learning enhances the learning process of machine learning algorithms.

Yasarka at el [78] proposed a method that used TL for image draining. They mixed the labeled data with unlabeled data. The proposed algorithm learned the network parameters from the mixed data. Then it transferred the learned parameters to the target domain where the data is not labeled. They evaluated their algorithm using three data sets. The results showed that this algorithm overcomes existing methods. They suggested the using real data in the training process would result in better performance. They also showed that pictures they took in rainy weather can be visualized more accurately using their algorithm.

Talukdar at el [64] implemented TL with computer vision field. They overcome the problem of preparing training data, by using full synthetically generated data. They analyzed the performance of TL and CNN on a real data set and online data that has never been used in the training process. The authors evaluated their study by transferring a single strategy of augmentation and a combination of augmentation strategies. They conclude that transferring a combination of augmented data strategies, for example, random-Corp, pixel shift, and rotate to the target domain will increase the performance and accuracy of the image detection.

Alzubaidi et al [3] implemented TL with Medical image analysis. Their study was concerned with detecting different cancer images. They proposed a new approach to TL. They trained and transferred the knowledge from large unlabeled data to train the deep learning model on a small set of labeled images. They constructed a mixed deep convolutional neural network (DCNN) model. This model gathers several ideas, for example, layers of parallel convolutional,

and residual connections. The authors introduced a new type of transfer learning called double transfer learning. This type of transfer knowledge consists of many steps:

- Step 1: train the model from scratch on a dataset.
- Step 2: use a trained model and train it on the same dataset from step 1.
- Step 3: tune the trained model from step 2 and train it on another dataset.

The authors evaluated their work using more than 200,000 cancer medical images. The results showed that TL can improve the learning process that results in high prediction and classification.

Wu et al [75] implemented TL with machine-bearing fault diagnosis. They proposed a method that was constructed of three parts. They built the model using the recurrent neural network Long Short-Term Memory (LSTM) to learn the relationships between the source and the target domains and to generate a dataset. The Joint Distribution Adaptation (JDA) learned the classification parameters by adjusting the differences between the conditional probability and marginal probability of the generated dataset and the target domain data. After the JDA learned the classification parameters the Gray Wolf Optimization algorithm was applied to optimize the JDA parameters. The authors confirmed the performance of the proposed method by experimenting with two different datasets and a small amount of labeled data. They found out that using TL in the diagnosis process was a powerful tool. The results showed this method achieved high diagnosis results compared with other diagnostic methods.

Xu et al [77] implemented TL in the field of subwavelength electromagnetics. They used TL and GA to design an effective dielectric metasurface model. They transferred the knowledge from small, and similar datasets to the target task. The authors proved the effectiveness of the proposed model by generating two deflectors and metalenses. The results showed that the forward spectrum-prediction neural network (FPN) learned the relationships between the meta-atoms and electromagnetic representation of the source domain by implementing TL faster, which helps to improve the performance and accuracy of the proposed model.

3.1 Transfer learning and Evolutionary Algorithms

Koçer and Arslan [35] implemented TL in GA. They aimed to enhance the performance of GA by using TL. The authors created an array that held the best individual and another chosen

individual of each iteration of the source population. They transferred 30% of this array to the transferred population and the remaining 70% of this array was created randomly. Then they used the transferred population as the starting point of solving the target problem. The results showed TL method improved the GA performance compared to the classic GA method.

Gupta and Ong [28] have studied the relationship between genetic transfer and population diversification. They aimed to discover what they called "the secret ingredients of evolutionary multitasking optimization". The authors used the Sudoku puzzles game as the case study and to evaluate their work. They provided empirical experiments and concluded the genetic transfer and population diversification are two sides of the same coin. This indicates genetic transfer and population diversification are important operators of Transfer Learning.

Muller et. al [47] have studied TL and genetic programming(GP). They proposed an algorithm that extracted the source problem sub-tree (blocks) to the target problem. They evaluated their proposed algorithm by solving symbolical regression problems. The results showed their proposed algorithm achieved high performance compared to classic GP and other GP methods.

Runa et. al [56] have studied TL in Dynamic Multi-Objective (DMO) environment. Their study focused on answering when to transfer and what to transfer. They proposed a method that has improved the performance of an algorithm called the Transfer learning-based Dynamic Multi-Objective optimization algorithm (Tr-DMOEAs). Surprisingly, the authors found that TL performed badly if the changes in the DMO environment were small. Their proposed method combined the following two steps:

1. Using the liner kernel function.
2. Selecting initial population from combined transferred solutions and copied solutions from previous environments.

The results showed that the proposed method was an efficient method of solving DMO problems.

Chapter 4

An Evolutionary Computation Based Model for Testing Transfer Learning Strategies

This chapter is exactly as published at the Congress on Evolutionary Computation CEC-2021 conference [2]. It addressed the first objective, which is "Compare five different strategies of transferred population". These strategies include the Estimation of Distribution (ED) concept, and this paper answered the following questions:

Q.1 : What are the properties of an effective collection of information to transfer?

Q.2 : How does the relationship between the source and target problem affect what information to transfer?

4.1 Abstract

To study how Transfer Learning (TL) works and what are effective strategies for transfer learning, we propose to model the TL process using Evolutionary Computation. EC provides a clear model for a problem as searching through a set of potential solutions. We are able to more easily control and measure problem difficulty, problem similarity, and methods of information transfer and relate these to success. As a proof of concept, we will use a static source problem and three fixed target problems with simple known relationships (see Section 8.4). We compare the effectiveness of several ways to transfer knowledge learned from solving one problem to solving the new problems in the context of the relationship between the problems. This we hope will demonstrate that using our EC model is a fruitful way to investigate TL. The results show there is an improvement for using some sampled methods representing the "learned knowledge" of the source problem S. Also, the results show that the diversity of the transferred population

has some positive effect on finding the optimal solution depending on the relationship between source and target problems.

4.2 Introduction

Transfer Learning (TL) [53] is using information from the solution of one set of problems to aid in the solution of similar new set of problems. This can speed the solution of the new problems. Also, by considering a broader range of example problems that don't have to be identical, it may provide a richer source of training data for Machine Learning.

TL deals with two problems: the source problem, S, and the new target problem, T. We consider the simplified case of a single source and target problem. Transfer Learning strategies aim to transfer information gained in solving S to solving T. This may reduce the time and effort of finding a solution of the new target problem T [53] [69]. The effectiveness of using the information from S is highly dependent on the relationship between S and T. If S and T are very similar, the information from solving S may be very applicable. If not, then the information may actually mislead and slow the solution of T. This is sometimes referred to as negative transfer learning.

Our goal is to study how TL works. But to do this effectively, we need to measure the difficulty of source and target problems and create a measure of how and to what degree the source and target problem differ. Furthermore, we need a way to characterize in a measurable way the information that is transferred and its affect on the search for a solution to the target problem. Finally, we need an algorithm that is powerful but simple in its architecture as a means to study the affect of the TL. In this paper we satisfy these requirements by prototyping some relatively simple source and target problems defined over bitstrings with clearly defined and measurable differences. We have chosen EC as the optimization algorithm because it solves the problems using a transparent, simple, and effective search. Also with EC we can use the initial populations as a natural way to characterize the information transferee.

The study of TL has broader impacts in that, TL mimics how humans often approach problems. Usually, when a person faces a new problem, they will think about a similar situation they have faced before, and, based on that experience, they will strategize about how to solve the new one. The relationship between the new problem and the old one may or may not be similar enough to be useful.

To study how TL works and what are effective strategies for transfer learning, we propose

to model the process using Evolutionary Computation (EC) [20]. Evolution provides a clear model for a problem as a search through a set of potential solutions. With our approach we will be able to more easily control and measure problem difficulty, problem similarity, and methods of information transfer. As a proof of concept, we will use one static source problem and three fixed target problems with simple known relationships (see Section 8.4). We compare the effectiveness of several ways to transfer knowledge learned from solving one problem to solving the new problems in the context of the relationship between the problems. This we hope will demonstrate this is a fruitful way to investigate TL.

We limit ourselves to two questions:

- **Q1:** What are the properties of an effective collection of information to transfer?
- **Q2:** How does the relationship between the source and target problem effect what information to transfer?

We will show how to reframe these questions in terms of an evolutionary algorithm and give some very specific problems.

4.3 Background

In this section, we briefly review related work in Transfer Learning, system diversity, and evolutionary computation.

4.3.1 Transfer Learning

Researchers have used a variety of methods to transfer knowledge from source to target solvers. Often they aren't concerned with the general question of how to do transfer learning, but use TL in a practical problem. The Meta-Learning Method is one of the machine learning methods that have been implemented with TL. Meta-learning aims to gather data or knowledge from multiple sources and tasks. Sun et al. [63] proposed a novel method where they exploited the advantages of TL and meta-learning together in one model called Meta-Transfer Learning (MTL). They used MTL to train a deep neural network (DNN). The aim of their model is to help the DNN to converge faster while using fewer labeled training data or weights. They trained DNN weights on large scale data. The MTL model not only transferred weights, but learned and transferred the operations of scaling and shifting on the parameters.

Transfer Learning has been used in "urban computing" to predict air quality [73]. The results showed a significant improvement in the air quality prediction. The authors addressed

the following question: "Can we transfer knowledge from a city where data are sufficient, to a city which faces either label scarcity or the data insufficiency problem?". The authors proposed the Flexible multi-mOdal tRAnsfer Learning (FLORAL) method. They claimed this method is flexible enough to transfer data from multiple sources and overcome the following challenges by using transfer learning:

- Transfer multi-modal data between the source and the target problem.
- Ameliorate the insufficient data problem.
- Predict the air quality in three cities, with high performance.

Transfer Learning has show great results in image classification. Long et al. [40] proposed the Graphic Co-Regularized Transfer Learning (GTL) framework, which addressed two problems of transfer learning: ineffective transfer and negative transfer. The ineffective transfer is a situation where the transferred data is **underfitting** the target problem needs. They claimed this situation can be overcome by maintaining the geometric and statistical probabilities together of the original data in the source problem. In contrast, the authors described negative transfer as a situation where the transferred data is **overfitting** the target problem needs. For this situation, they maintained the geometric structure of the source and the target problems. They claim if the geometric structure of the target problem contradicts the source, geometric structure of the target problem will be respected. They defined the geometric structure as the embedded manifold. The authors transferred popular latent factors of the source data. These factors are statistical properties of the original data or points across both problems. These data are optimized again by maintaining the geometric property of the target domain. The results of their work can be summarized in the following points:

- GTL in a unified framework maintains the statistical property and geometric structure together
- They proposed two new methods that help for cross-problem text and image classification.
- The experiments were done on both text (Reuters-21578 and 20-Newsgroup) and image (PIE, USPS, MNIST, MSRC, and VOC2007) and the results show how GTL has a positive effect.

4.3.2 Diversity and Entropy

Diversity of a population is important to maintain effective exploration of the solution space by an evolutionary algorithm. Diversity can be measured by entropy.

In S.K. Smit and A.E. Eiben [60], they tuned the parameters of the Evolutionary Algorithm (EA) to achieve high performance. They claimed in their paper entropy can measure the relevance of EA parameters. They described an entropy-based algorithm called REVAC (Relevance Estimation and Value Calibration) which uses the entropy to estimate the EA parameters. The results showed great improvements in performance.

Vargas et al. [70] have used wind and earthquake time series to initialize a GA's population. The result of their study shows an improvement in GA performance by using specific distributions and diversity in initial population generation. This has implications for initial populations for target problems in TL.

Previous work has demonstrated that diversity is an important factor in EAs. Our work here will provide a case study of using diversity of the transferred population to improve TL performance. We believe a carefully constructed diverse population will aid in the solution of the target problem. We hope our findings provide a way of improving Transfer Learning for solving hard problems. We will express these ideas more formally in our hypothesizes to follow.

4.3.3 Evolutionary Computation

Evolutionary Computation (EC) is the study of algorithms that use Darwinian like processes of population, reproduction, and selection to solve optimization problems [12, 20, 26]. As such, EC borrows many of its terms from evolutionary biology. EC is a black box optimizer used for solving problems in which no simple algorithm for solving that specific kind of problem may be known. It has been shown to be highly successful at solving a wide variety of practical problems. EC is an ever-expanding field and has many sub-fields like genetic programming, genetic algorithms, evolution strategies, and others [6, 20].

Genetic Algorithm's (GA) [46] are a subset of algorithms in EC. GAs are general optimization algorithms in EC that generally, but not not necessarily, work on fixed length linear data structure called a gene that represents a potential solution to a given problem. A population is a set of genes.

In a repeated repeated process, Genes with higher fitness are preferentially selected. The selected genes are replicated with variation and mixing between genes to create new genes in

a new population. And the new population replaces the old one. In a generational GA, an process of selection and replication with variation is referred to as a **generation**. In repeated applications of selection and replication (generations) the population is said to converge to genes with better fitness.

In general, the population will explore regions of the fitness space with higher fitness. The variation in the genes is caused by two kinds of operation: mutation and crossover. Mutation of a gene causes small random variations in the gene which will likely create a gene with similar fitness to the original. That is, the mutation operator creates genes of correlated fitness. The classic crossover operator is a mixing operator that randomly assembles a gene from two genes. This assembly process may accidently assemble compatible parts that yield new higher fitnesses. The variation in the genes allow the algorithm to **explore** the fitness space. The selection of genes of high fitness **exploit** the structure of the fitness space. We will use a version of GAs in our experiments because of its algorithmic simplicity and transparency of exploration.

4.3.4 Transfer Learning and Evolutionary Computation

TL has been implemented in many ML algorithms. [53] [74] review the use of a variety of TL approaches in many real-life applications. They conclude that the benefit of using TL is not limited to just speeding up the process of finding a solution, but includes a flexible approach to expanding the set of usable training data by adapting multiple sources of data to solving a target problem.

[35] implemented TL in combination with GA. They enhanced GA performance by using TL. The authors created an array called "individual pool" which consisted of the best individual and a random good individual from the source population of each generation. Then they created a transfer population by transferring 30% from this pool and 70% was created randomly. The result showed this method improved the performance of the GA compared to the classic genetic algorithm performance.

Gupta and Ong [28] studied genetic transfer and population diversity. They discussed relationship between genetic transfer and population diversity. The authors used the Sudoku puzzle to evaluate their algorithms. They conclude that both genetic transfer and population diversification are important factors in solving Multitasking problems.

In general, TL consists of three main questions: what to transfer, when to transfer, and how to transfer. Runa et al [56] studied the last two questions of TL in the Dynamic Multi-objective

Optimization (DMO) environment. They surprisingly concluded that TL would perform worse if the changes between the source and the target are small and when the Pareto Optimal Set (POS) is fixed. They proposed a method that improves the performance of the Transfer Learning-Based Dynamic Multi-Objective Optimization (Tr-DMOEA) algorithm in solving DMO problems. The result showed the proposed method was more efficient than other recent methods.

Muller et al [47] studied TL in Genetic Programming (GP) using symbolic regression problems. They proposed a method that extracted blocks from the source problem to the target problem. The proposed algorithm was evaluated by comparing against classic GP and other recent GP methods. The results showed the proposed method was more effective than compared methods.

4.4 Methods

Evolutionary computation was chosen for this work because it provides a clear clean and measurable model of the process of transfer learning, allowing us to study many aspects of TL. It gives us a general model of problem solving as search through a space of potential solutions. It is easy to follow what information is known about a problem during solution by doing statistical analysis of the the population content. The problem difficulty of our test problems can be controlled by managing fitness and representation of the gene of the individual in the context of our Evolutionary computation model. This controls the "shape" of the fitness landscape. Problem similarity can be measured by comparing the fitness functions with respect to the representation. Finally, the knowledge transfer from source to target problem can be simplified to how is the population for the target problem initialized based on information gleaned in solving the source problem.

Our experiments will look only at cases where the target problem requires that the solver learn some new information in order to solve the problem. Our fitness will be simple in that it is just a reward for learning that new information. By changing this fitness we can study many more situations in which information, for example, becomes useless or even hinders reward. We can also easily examine problems that are misleading.

Specifically for our experiments, we will use a static source problem S and fixed target problems T. This model can be used to study and measure what is the effect of adding new knowledge requirements to T. We consider whether this addition of the new knowledge increases the number of generations the GA solver must take to find the optimal solution. The number

of generations is proportional to the number of probes into the fitness landscape and hence how long it takes or how much information must be gathered to solve the problem. This way, we can study what effort the TL must apply to solve the problems.

In our model, we first solve the problem S using a GA starting from a randomly initialized population. Then we transfer information in the form of genes discovered in solving S into a new population and use a GA to solve T. This way our model blends together the Transfer Learning (TL) strategies and Genetic Algorithms (GA).

For our experiments, our GA is a generational GA that will work on a gene consisting of 40 bits (this is related to the fitness function will follow). Selection uses a tournament selection in which the best of k randomly chosen genes is selected for the right to reproduce. In our case, $k = 3$. Further details of the GA parameters [20] are specified in Table 8.1.

Table 4.1: Genetic Algorithm Parameters

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	100
Mutation rate (per bit)	0.1
Crossover rate	0.01
Type of crossover	Uniform crossover
Tournament Size	3

The **fitness function** maps a gene to a real value that measures the quality of the gene. The GA will try to maximize or minimize whatever quality the fitness function measures. Our function is defined on a number of bits and is composed of sub-functions, similar to an idea of Embedded Functions [29]. We will try to maximize the fitness function to achieve an optimal solution. We defined the fitness function over n bit space: $f : \mathcal{B}^n \rightarrow \mathbb{R}$:

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1]) \tag{4.1}$$

where subfunctions $g_i : \mathcal{B}^s \rightarrow \mathbb{R}$ are defined over s bit subsections of the gene x. There are m non-overlapping (in this case) subsections and m subfunctions. Let $n = m * s$. Let $a_i \in \mathbb{R}$. For $x \in \mathcal{B}^n$, and $x[a, b]$ extracts bits in positions a through b from bit string x. In our experiments: $s = 4, m = 10, n = 40$ and $a_i \in \{0, 1\}$.

For our experiments, a_i is limited to 0 or 1 to model the importance of each g_i to the solution of the problem. If $a_i = 0$ then the g_i need not be solved at all. If $a_i = 1$ then solving g_i contributes to the fitness and become important. It is like having m potential skills to be

learned but only the ones with $a_i = 1$ are really needed. Varying the vector $\vec{a}$ selects what skills or knowledge is important.

Our model finds the optimal solution by solving sub-functions. Each individual is divided into 10 sub-functions (g_i). In order to get the optimal solution, some these sub-functions must be solved. The functions we will use for g_i are called a **Deceptive Functions**. This function is relatively hard to solve. A deceptive function improves as the number of 0 bits in its argument increases, but the actual best value is with all 1 bits (see (7.2b)). This structure is deliberately misleading.

$$g(b) = \begin{cases} s & \text{bc}(b) = s \\ s - 1 - \text{bc}(b) & \text{otherwise} \end{cases} \tag{4.2}$$

where the bit count function $bc(b), b \in \mathcal{B}^s$, is defined as the number 1 bits in b. Algorithms are deceptively lead to solutions with all 0's however, the best answer is at all 1's.

As a proof of concept we pick a single source problem S given by it's $\vec{a}$:

$$\vec{a_s} = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0).$$

This gives us more control of the difficulty of the problem. We can compare different situations of transferring knowledge. For example, what is different if the T problem is different from the S problem by a small number like adding one new nonzero element to $\vec{a_s}$ to get $\vec{a_t}$? What is the effect of having a larger difference between $\vec{a_s}$ to get $\vec{a_t}$?

We choose **three target problems** (7.2ca, 7.2cb, 7.2cc) for our study: the first one adds one new element to solve the target problem. The second adds four new elements to solve the problem. The third one adds eight new elements to solve the target problem.

$$\vec{a_t} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{4.3a}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0) \tag{4.3b}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{4.3c}$$

We hypothesize that the diversity of the transferred population is an important indicator of the success of transferring knowledge in Transfer Learning (TL). This is because diversity will improve exploration of the solution space. We also hypothesize that the content of the transferred population is more important the more S and T are similar. This is because this controls the exploitation of the previous solution. These two hypothesizes are directly related to two questions in our introduction.

4.5 Experiments

4.5.1 Experiment 1

In our experiment we solve the source problem using a GA denoted GA(S). We then look at five strategies for sampling the population from the source problem solution to create a **transferred population**. The transferred population is then used as the starting point for solving the target problem. We include a direct solution of the target problem from a random population as a control (Fig. 4.1). We measured the diversity of each transferred population (see (4.5)). Also, we counted how many generations the target problem solver (GA(T)) took to find the optimal solution. 50 replicates were run for statistical analysis for each sampling technique and target problem.

Our five different transferred populations are labeled: ED, 30-Top, 30-Best, 100-Top, and 100-Best. Our control is labeled simply: GA. It is a copy of the initial population that passed to the GA(S) (see bottom of Fig. 4.1).

The algorithm we use is a simple generational genetic algorithm with a 40 bit gene and a population of 100 see Table 8.1. The entire algorithm ensemble is run for each of the three target problems in (7.2c).

Details of the sampling strategies for creating the transferred population:

- **ED** Estimation of Distribution sampling uses the distribution statistics for each bit position in the population to generate the corresponding bit in the child gene. This idea stems from Estimation of Distribution Algorithms (EDAs) [36]. This samples the GA(S)'s population based on an (ED) strategy.

- **30-Top** The top or best 30% of the GA(S)'s population is copied and the remaining 70% of the population members are generated randomly.

- **30-Best** The best individual of the GA(S)'s population is copied into 30% of the population. The remaining 70% is generated randomly.

- **100-Top** An identical copy of the S's final population is used. This can be thought of as copying the top 100% of the population, hence the name.

- **100-Best** The best individual of the GA(S)'s population is copied into 100% of the population.

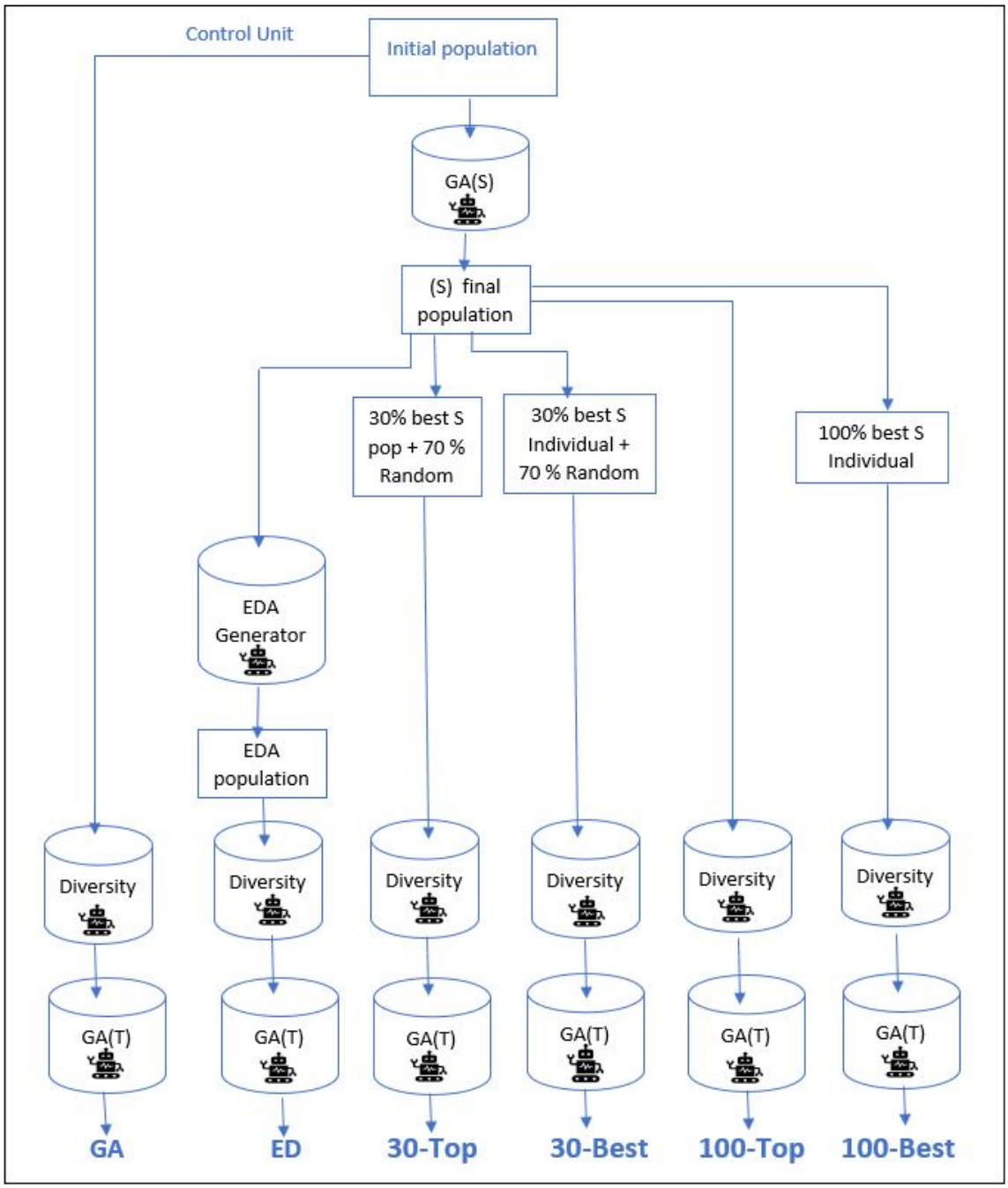

Figure 4.1: A randomly initialized population is created and passed to the GA(S) to find the optimal solution to source problem S. A copy of the initial population is passed to GA(T) as a control. The final population is then sampled five different ways: an ED sampler based on Estimation of Distribution algorithms, top 30% of the population with the remaining generated randomly, copying of the best individual to 30% of the population and the remaining is generated randomly, copy of 100% population (the whole population), copying the best individual to 100% of the population. The control and the five different transfer populations are each passed to the GA(T) solver to find the optimal solution.

Each TL strategy has its own approach to 1) capturing knowledge from the source problem and 2) maintaining diversity in the initial population for solving the target problem. 100-Top and 100-Best invest solely in the source problem population with 100-Top getting its diversity from the source problem population diversity. 100-Best has no diversity but relies on simply the best single answer. 30-Top and 30-Best get added diversity from randomizing 70% of the population while sampling from the best of the source problem. Finally, ED mimics the bit by bit distribution frequency of the source problem without selecting any specific solution. So the diversity is the same as the source problem but whole solutions are not preserved.

For each of the target problems, we have measured and plotted the diversity of the transferred population. Diversity was computed using an entropy method. **Entropy** from Information Theory measures the randomness in a sequence of symbols by looking at the information content of each symbol. If the next symbol in the sequence is more unpredictable then that symbol brings with it more information. If it is extremely predictable, then it has low information content. Usually, entropy of a discrete random variable X is denoted by $H(X)$. The entropy is calculated as follows: [58]:

$$H(X) = -\sum_{i=1}^{||X||} P(x_i) \log_b P(x_i) \tag{4.4}$$

where x_i is one of the n possible symbols in X and $P(y)$ is the probability of seeing y as the next sample from X. $||X||$ is the number of different symbols in X. In our study $||X|| = 2$ since X is the 0 or 1 values in the individuals. b is the base of the logarithm and controls the units for the measure of entropy. In our study $b = 2$ meaning our entropy is measured in bits. Maximum information (randomness) happens in our experiment when $H(X) = 1$ meaning each locus has an equal probability of being either a 0 or a 1. $H(X) = 0$ happens when all the symbols in X are the same.

A population, P, consists of a set of N individuals each of which has n loci as described in the fitness function above. The values in a given position in the population $P[i]$ can be thought of as a random variable with N sample values. The **diversity** $D(P)$ computed as the sum of entropy across all loci or positions in the gene. That is, $H(P[i])$ uses the values in position i across all individuals in the population as the random variable.

$$D(P) = \sum_{i=1}^{n} H(P[i]) \tag{4.5}$$

We computed the diversity of the transferred population for the three different target problems (see Section 8.4). The diversity of all three target problems were similar to Fig. 4.2. The 100-Best population diversity was less diverse than all other population diversity in all three T problems. The reason why the 100-Best population is less diverse is that the component of the population. All population of the 100-Best is a copy of one individual (the best individual of the S's problem). This is why the 100-Best's shows less diversity in all experiments.

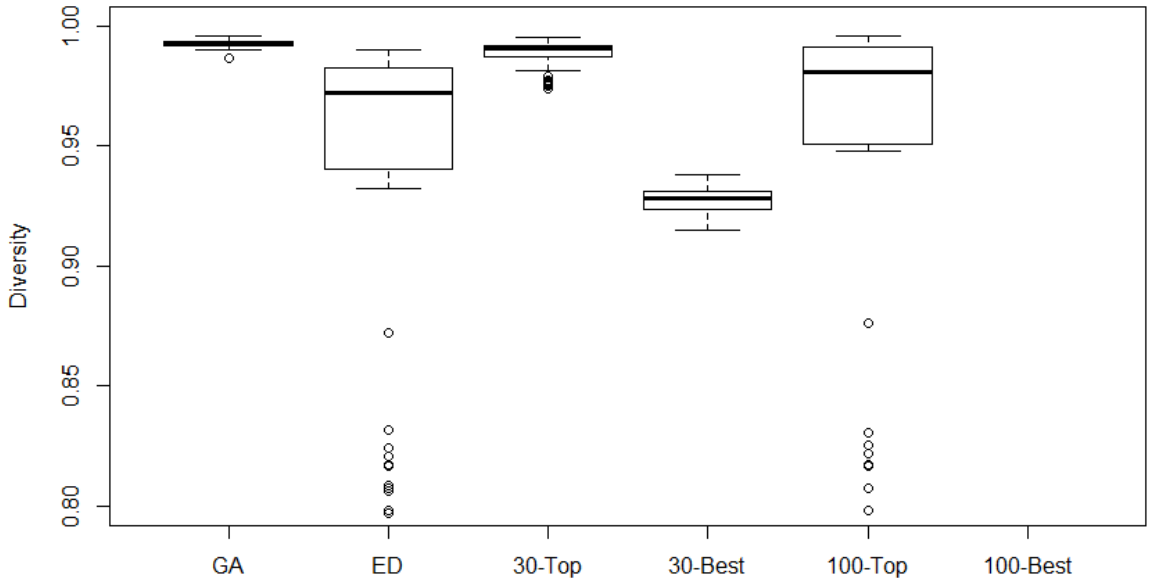

Figure 4.2: This diagram is the diversity over 50 samples of each population type as measured using (4.5) equation and shown as quartiles. Because it uses entropy as its measure of diversity a measure of 1 is random and 0 is all identical. In this diagram, the GA(T) solver must add one new element to solve the problem T. The 100-Best's population is less diverse than the other five populations. The 100-Best's diversity was equal to 0.0 in all experiments and so is not included in the graph. 100-Best's population is 0 because the population is just copies of the best individual of the GA(S)'s population.

Fig. 4.2 shows the distribution of diversity of the population for each strategy as quartiles. The population diversity was measured before the population passed to the GA(T) solver. We can see that GA's, 30-Top's, and 100-Top's population have more diversity than the ED's, 30-Best's and 100-Best's population. This experiment was run using the first target T problem which is adding one new element. But all three problems show similar distributions.

4.5.1.1 Target Problem 1

The first target problem (7.2ca) is one in which the solver must add one new subfunction to source problem solution to solve the target problem. Fig. 4.3 shows how many generations it takes the GA's solver to find the optimal solution for each transferred population.

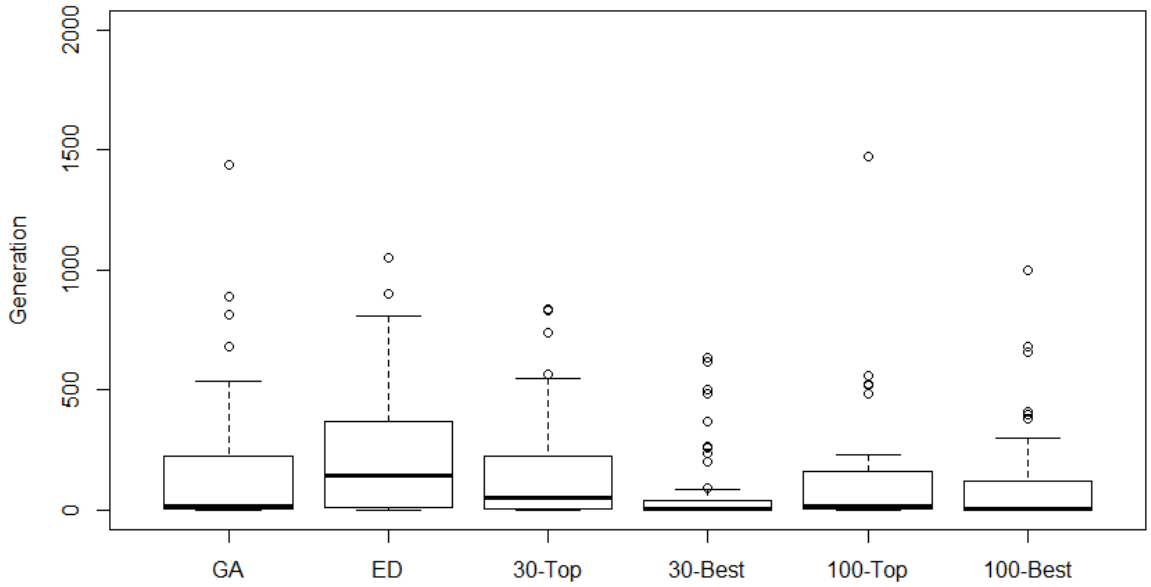

Figure 4.3: The number of generations to solve target problem 1 for each population type shown as quartiles. 100-Best and 30-Best's transfer strategies consistently take fewer generations than the other five populations. In this diagram, the GA(T) solver must add one new element to solve the T problem.

Since the distribution of generations-to-solution is not normally distributed, we ran the pairwise Mann-Whitney U test, which is a distribution free multiple comparison test, of the five sampled strategies of the transferred population, as shown in Table 4.2. We ran the Wilcoxon Rank Sum Test between GA and 30-Best and got a p-value equal to 1.04505. That is enough statistical evidence for the improvement by using Transfer Learning and the 30-Best strategy.

Table 4.2 shows which transferred populations are different from each other on the first target T problem. 100-Best and 30-Best significantly out-performed all other transfer strategies. The significant relationships between the transfer learning strategies is illustrated in Fig. 4.4.

Table 4.2: This is the Pairwise Mann-Whitney U test results of the first target problem, for the target problem 1 we use a pairwise Mann-Whitney U test to test if differences in performance are significant. The Mann-Whitney U test yields p-values. For our test, p-values < 0.05 indicate the median values of number of generations to find the optima for the pair of strategies is probably different. The table shows whether each for the strategies of sampling the population yields a different solution time from the others. The bold values of the table indicate where there is a significant improvement in finding the optimal solution and the arrow points at the best strategy. The median number of generations to solution in the control GA was 15 and the median performance of each transfer strategy is shown in the last row.

	100-Best	100-Top	30-Best	30-Top	ED
100-Top	↑**0.023**	-	-	-	-
30-Best	0.720	←**0.0003**	-	-	-
30-Top	↑**0.004**	0.355	↑**0.0001**	-	-
ED	↑**5.905**	↑**0.004**	↑**2.407**	0.067	-
GA	↑**0.002**	0.271	↑**1.005**	0.887	←**0.043**
Generations	3	14.5	2	50.5	142

4.5.1.2 Target Problem 2

The second target problem (7.2cb) is one in which the solver must add four new elements or subfunctions to source problem solution to solve the target problem. Fig. 4.5 shows how many generations it takes GA(T) to find the optimal solution.

We ran the Wilcoxon rank sum test between GA and 30-Best results with a p-value equal to 0.037. That is enough statistical evidence for that 30-Best sampled population out performs the control. Also, we ran the pairwise Mann-Whitney U test, which is a multiple comparison test of the five sampled population passed strategies, as shown in Table 4.3.

Table 4.3: The pairwise Mann-Whitney U-tests for target problem 2. A low p-value, p-values < 0.05, indicates which distributions of time to solution are different. The GA(T) solver must add four new elements to solve the T problem. The bold values of the table show (p-value < 0.05) where there is a significant improvement in finding the optimal solution. The median number of generations to solution in the control GA was 155 and the median performance of each transfer strategy is shown in the last row.

	100-Best	100-Top	30-Best	30-Top	ED
100-Top	↑**0.027**	-	-	-	-
30-Best	0.772	←**0.005**	-	-	-
30-Top	↑**0.014**	0.906	↑**0.013**	-	-
ED	↑**0.021**	0.923	↑**0.009**	0.825	-
GA	0.061	0.523	↑**0.037**	0.646	0.510
Generations	123.5	156	100.5	187	164

Table 4.3 shows how each strategy of the transfer population differs from each other when attempting to solve the target problem. Each value of the table indicates, by a p-value, how each sampled strategy is different from the other strategy. The bold values show where there is a

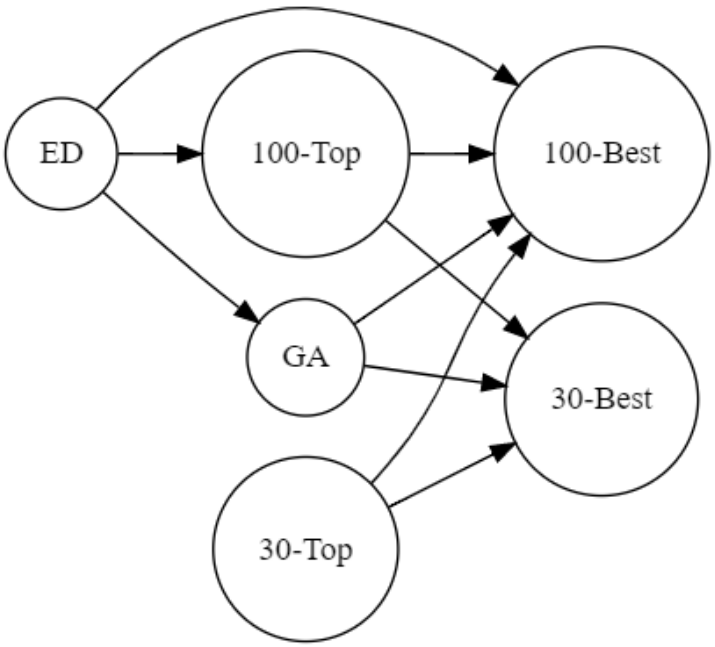

Figure 4.4: A partial ordering by significance of performance for different transfer strategies in solving target problem 1. Faster solution strategies are to the right in the graph.

significant improvement or difference between the transfer learning strategy's solution efficiency. The other values show how different each sampled population is, but these differences are not significant. The 30-Best and 100-Best strategies shows a significant improvement compared to the other strategies.

4.5.1.3 Target Problem 3

The third target problem (7.2cc) is one in which the solver must add eight new elements to source problem to solve the target problem. Fig. 4.7 shows how many generations it takes GA(T) solver to find the optimal solution for each strategy for the transfer population.

We ran the pairwise Mann-Whitney U test comparison test of the five transfer strategies and the control labeled GA. The results are shown as the p-values in Table 4.4. The bold values show where there is a significant difference between the transfer learning strategy's solution efficiency. The 30-Best and the 30-Top strategies showed a significant improvement compared to the other strategies. The pairwise significance between the strategies is illustrated as a partial ordering graph in Fig. 4.8.

The performance of the difference strategies do not vary much. This is because the GA(T) must add eight new elements to solve problem T and there is little information from the source

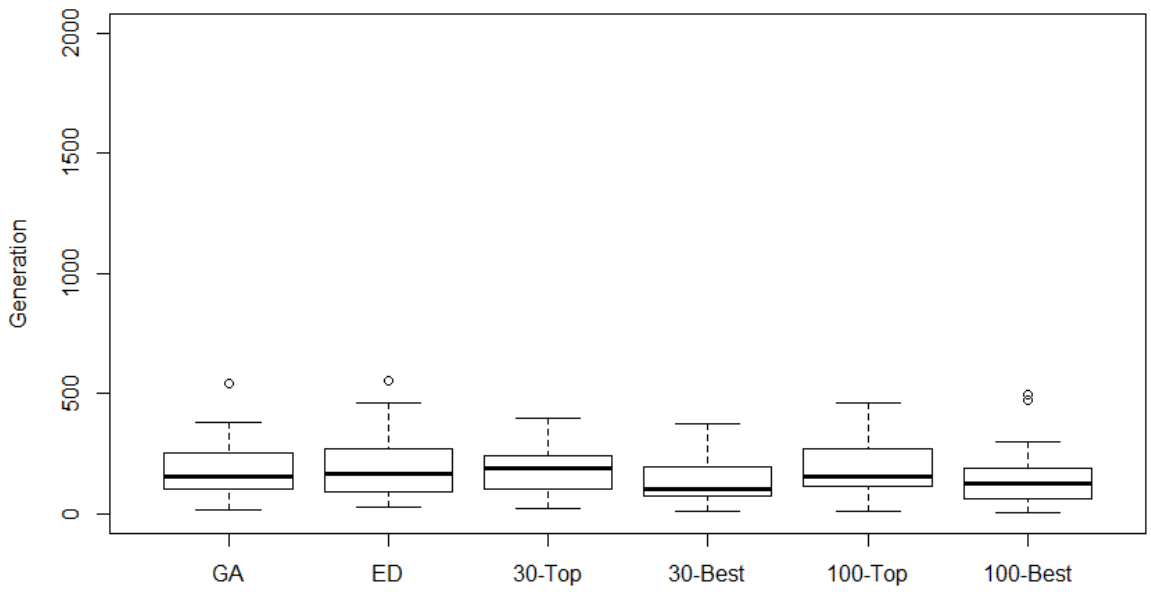

Figure 4.5: Number of generations to solve the target problem for each population type. The 30-Best's population takes fewer generations than the other five populations type.

problem to help solve this greatly different problem! That is, the T problem in this situation is totally different from the original S problem. Even though the T problem was more complex. The 30-Best and 30-Top strategies perform slightly but significantly better than the other strategies.

4.5.1.4 Discussion

Different strategies for transferred populations have an effect on the GA(T) performance of finding the optimal solution. Even though every sampled strategy used the same S's population of sampled the transferred population. The amount and character of transferred knowledge was different among these transferred populations. The 30-Best strategy of transferred population consistently outperforms the other strategies for the three problems, even when the T problem was harder as in the third T problem.

As we can see, the diversity of transferred population alone is not enough or the only criteria to determine how effective the TL system will be. This experiment shows there are other critical elements to determine performance of a Transfer Learning strategy.

There are two main factors: Learned Knowledge and Diversity. Some transferred popula-

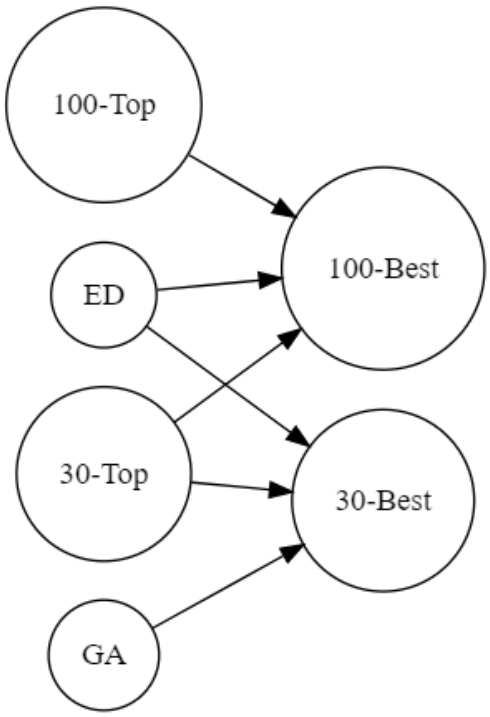

Figure 4.6: A partial ordering by significance of performance for different transfer strategies in solving target problem 2. Faster solution are to the right in the graph.

tions were more diverse than others (Fig. 4.2). Some populations incorporated what might be thought of as more knowledge of the source problem, for good or bad.

So, why were 30-Best, 100-Best and 30-Top (in some cases) faster at finding the optimal solution? We can analyze the two main elements in the transferred population.

As seen in Fig. 4.3 through 4.8, the 30-Best took fewer generations than the other strategies to find the optimal solution, even though the 30-Best's diversity was significantly less than the other sampling methods (See Fig. 4.2). 100-Best performed well for problems 1 and 2 but not as well for problem 3. One might imagine that as the problem differed more and more from the source problem, the best individual in the source problem became less relevant. 30-Top invests in more diversity it performed better on target problem 3 while lost to 30-Best in the the other two problems. The bet in higher diversity might have paid off in the problem most distant from the source problem.

We have shown that the right combination of knowledge and diversity is necessary for transfer learning to show statistically significant performance improvement.

As we move from target problem 1 to target problem 3, the problems become harder and

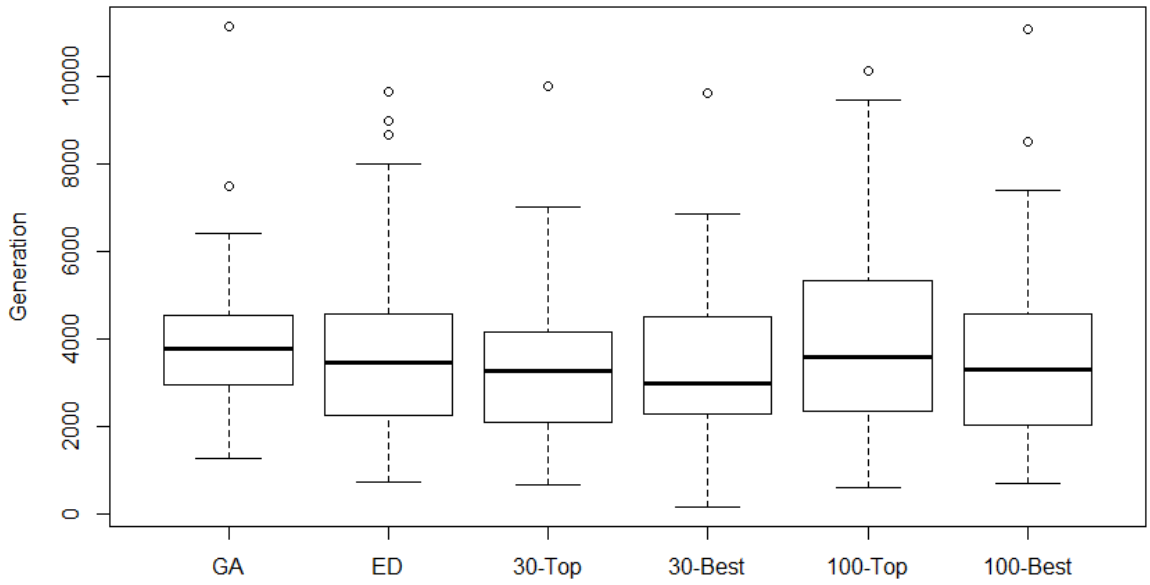

Figure 4.7: The number generations of each transfer population type. The 30-Best and 30-Top transfer populations take fewer generations than the other strategies.

we must pay more attention to the diversity of the transferred population. In the first two problems, the transferred population diversity did not play as important a roles in speeding up the process of solving the target problem. The 100-Best strategy has 0 diversity and yet was one of the significant strategies for finding the optimal solution faster in problems 1 and 2. But when the problem became harder like the third problem, transferred population diversity became an important factor. This is why the 30-Top strategy overcomes the 100-Best strategy in the third problem. This analysis leads us to study transferred population diversity in the second experiment.

4.5.2 Experiment 2

The aim of the second experiment is to understand the role of diversity in transfer population on performance. To do this we select one approach which is picking the best individual from the final population of the source problem. We then vary the amount of random individuals in the initial population for solving the target problem to control diversity. The second experiment is essentially a comparison of three similar sampled populations with differing diversity.

Table 4.4: The pairwise Mann-Whitney U-tests for target problem 3. The bold values of the table specify (p-value < 0.05) where there is a significant improvement in finding the optimal solution. The median number of generations to solution in the control GA was 3789 and the median performance of each transfer strategy is show in the last row.

	100-Best	100-Top	30-Best	30-Top	ED
100-Top	0.393	-	-	-	-
30-Best	0.920	0.224	-	-	-
30-Top	0.728	0.213	0.899	-	-
ED	0.705	0.654	0.612	0.400	-
GA	0.104	0.749	↑**0.047**	↑**0.050**	0.303
Generations	3291.5	3583	3017	3267.5	3468.5

- 30-Best: a combination of: the best individual from the S problem copied into 30% of the population and the remaining 70% a randomly generated population.
- 60-Best: a combination of: the best individual from the S problem copied into 60% of the population and the remaining 40% a randomly generated population.
- 100-Best: a combination of: the best individual from the S problem copied into 100% of the population and the remaining 0% a randomly generated population.

We ran the experiment 50 times on the second target problem, where GA(T) must add four new elements to solve the T problem. This is the problem of medium difficulty from our problem suite. Fig. 4.9 and 4.10 show the results.

Fig. 4.9 shows the diversity of 30-Best is more than the diversity of the 60-Best which is more than the diversity of the 100-Best population. This difference in diversity is because the 30-Best population has 60% random population on the other hand, the 60-Best population has 40% random populations and the 100-Best population has 0% random population.

Fig. 4.10 shows how many generations it takes the GA's solver to find the optimal solution for the transfer strategies 30-Best, 60-Best, and 100-Best.

We ran a Kruskal-Wallis test between 30-Best and 60-Best to see the difference between them. The p-value was equal to 0.05024. This is boarder line statistical evidence of the population diversity has an effect (p=0.05) of finding the optimal solution.

The second experiment shows the diversity of the transferred population has an effect on the performance of the GA solver in finding the optimal solution given that the best element from GA(S) is already in the initial population. All populations of the second experiment are sampled using the same source population from $GA(S)$ with different amounts or random

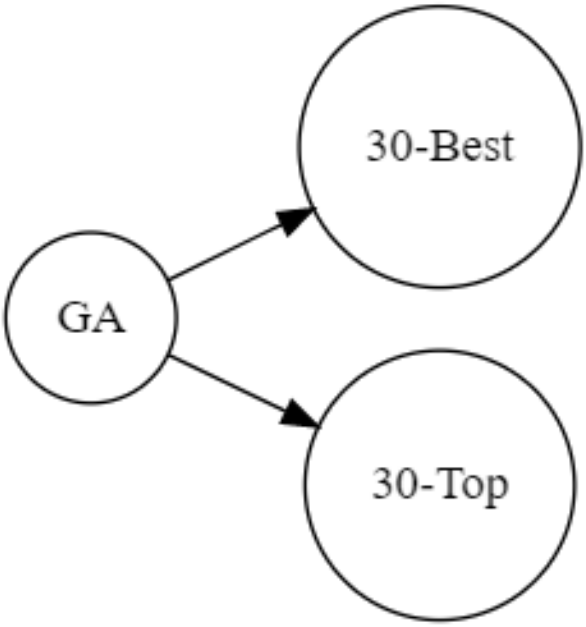

Figure 4.8: A partial ordering by significance of performance for different transfer strategies in solving target problem 3. Faster solution are to the right in the graph.

additional information. Their diversity is different, as shown in Fig. 4.9. It is more likely the GA can find the optimal solution easily with a more diverse population in addition to supplying just the best solution from GA(S). The strategy GA from experiment 1 can be thought of as 0-Best. We saw in experiment 1 that it performs poorly compared to 30-Best and 100-Best on problem 2 even though it has even more diversity that 30-Best. We believe this is because no information about the solution is available in the population. So our conclusions in the Experiment 2 assume that given the population contains solution to problem S then how does diversity help solve the problem.

4.6 Conclusion

From the first and second experiments, what we are transferring to the GA(T) to solve problem T is what really matters in the transferring population. How the population is sampled (the knowledge) and the diversity of the transferred population are both important. We have to tune the transferred population in a way to cover both the knowledge and diversity.

We ran two different experiments in order to answer our questions. The first experiment dealt with a spread of five different strategies of sampling for the transferred population. The second experiment was focused on the proportion of the population that was random versus

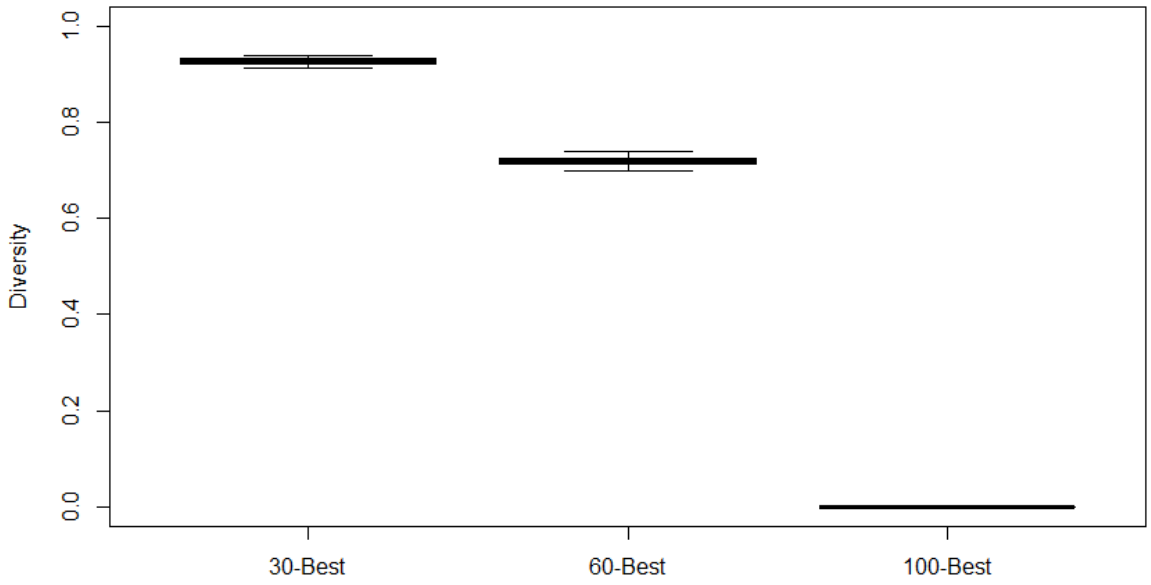

Figure 4.9: The population diversity of the 30% Best individuals of the GA(S)'s population, the diversity of the 60% Best individuals, and the diversity of 100% Best individuals. The graph shows the 30-Best population is more diverse than the 60-Best which is more diverse than the 100-Best population.

copies of the best. Both experiments measured the transferred population diversity and how many generations it took the GA(T) to find the optimal solution.

By analyzing the 30-Best population, we can see it is built on the classic exploitation and exploration [20]. Exploiting the known knowledge of how to solve the previous problem can be done by transferring parts of GA(S)'s final population. Exploration can be enhanced by adding the diversity to the transferred population. This idea allowed us to maintain what the problem has learned and allowed the solver to look more quickly for new optimal solutions. The 30-Best strategy of population sampling performs as well or better than all the other strategies discussed. Preliminary data suggests, as one would expect, this result is problem specific.

The main contribution of this work is an experimental design that incorporates easy to measure problem stucture and difficulty, a natural method of transferring information from one problem solution to another problem, and a relatively transparent problem solver with simple performance measures.

Now that we have found that our design has promise, we are expanding the classes of

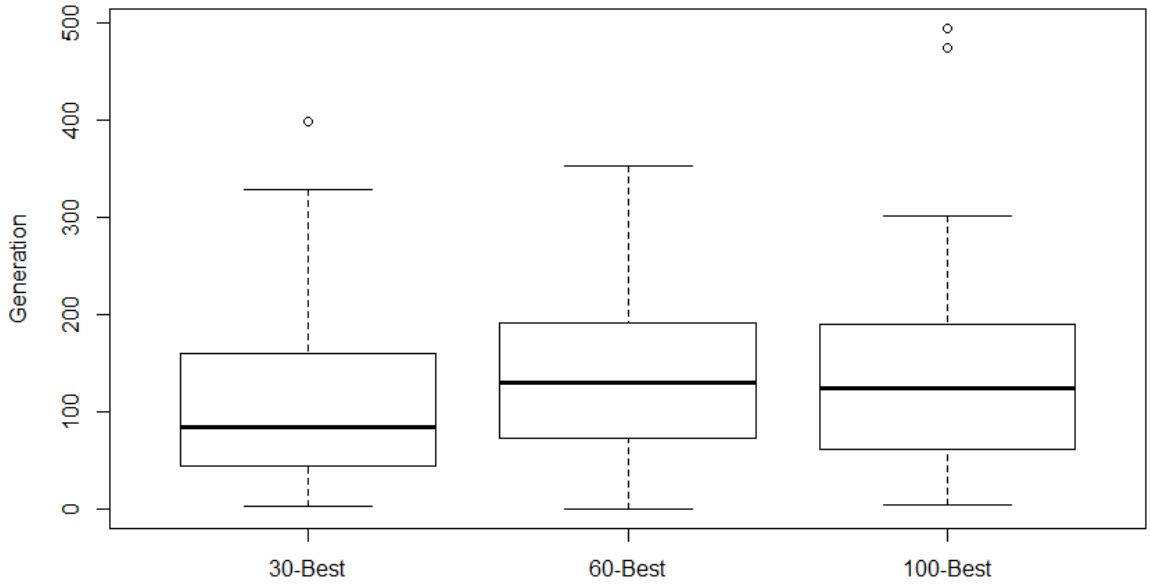

Figure 4.10: This graph shows how many generations the GA solver took to find the optimal solution to target problem 2. The 30-Best took the fewest number of generations.

functions to be optimized and performed a more broad spectrum of parameterized experiments in future work.

Acknowledgment

This work is supported by Saudi Arabian Cultural Mission and Albaha University, Albaha, Saudi Arabia. I would like to thank my supervisor Dr. Robert B. Heckendorn for his great advice in this study.

Chapter 5

Entropy-Based Algorithm of Transfer Learning and Genetic Algorithm

This chapter is exactly as published at the 2021 World Congress in Computer Science, Computer Engineering, & Applied Computing CSCE-2021 conference. It addressed the second objective, which is "propose an algorithm for sampling the transferred population". This chapter proposed an algorithm that sampled the transferred population from the S problem and transferred them to the T problem. The results of evaluated the proposed algorithm showed TL improved the performance of the GA compared to other strategies of TL. This paper answered the following questions:

Q.1 : Study which part of the S data has the most information about solving the problem?

Q.2 : Does transferring these parts of the S population help find the T problem's solution easily?

5.1 Abstract

Transfer Learning (TL) is a Machine Learning strategy that employs previously gained knowledge in solving one problem to solve a another similar problem or task. We created a model of TL using Genetic Algorithms (GA) to better understand TL. A Genetic Algorithm is a stochastic optimization method that searches and evolves candidates in a known environment looking for the best candidates. We proposed an algorithm that used the concept of entropy (H) to discover, maintain, and transfer known knowledge of a solved problem to solving another problem. The results demonstrate that our proposed algorithm is more efficient than some other TL strategies.

5.2 Introduction

Transfer Learning (TL) is a Machine Learning (ML) technique that has shown very promising results. Transfer Learning can reduce the time it takes to solve a new problem by leveraging information about a similar problem. In situations where there is not enough data, TL can also provide a strategy for combining data from multiple sources to solve a new problem.

TL deals with two problems: S and T. TL aims to use previously trained data from the source problem S and transfer it to the target problem T. TL should attempt to transfer the most useful information about the S problem with the hope that transferring these data will reduce the time and effort of finding the solution to the T problem. [53] [69].

The relationship between the S and T problems is an important factor in transferring data and in finding the optimal solution. If the problems S and T are similar, then the transfer process will reduce the time and effort of finding the optimal solution to the T problem. On the other hand, if the two problems are very different, then the transferring process of the data can make it more difficult to solve the T problem and increase the required time and effort. This situation of misleading information is called negative transfer [53].

Transfer Learning parallels how humans solve problems. Usually, when a person faces a problem, he/she will think about a similar situation he/she faced before. And based on that experience, he/she will develop a solution to the current problem. Similarly in transfer learning, we have data about solving a previous problem, and we transfer that data to assist in solving another problem. An important question for Transfer Learning is how to find the subset of data that is the most helpful in solving T. Answering this question will help TL designers find and quantify the core useful data of the S problem and transfer the data to the T problem. Also, answering this question may help to avoid the problem of negative transfer.

Our aim is to find a method that can discover which parts of the S data have the most useful information about solving the S problem, and to decide if transferring these data (partially or wholly) will help to solve the target problem T. Difficulties caused by negative transfer can be reduced or solved by discovering the most promising data and determining whether it will be useful to transfer.

To address this aim, we designed a general, tunable model of Transfer Learning using a Genetic Algorithm (GA) and system entropy (H) from information theory [58]. The GA will generate candidate solutions for both problems S and T to find the optimal solution. TL will

transfer the knowledge between the S and the T problem. The randomness of this model can be measured using entropy H. The entropy can relay how much each gene in the GA generated solutions 'knows' about solving the problem. This value can be used to determine if a particular gene or a bit of the data has sufficient information, or not, regarding solving the S problem to be useful to transfer.

The objective of this study is to answer the following two questions:

- **Q1:** Which part of the S data has the most information about solving the problem?
- **Q2:** Does transferring these parts of the S population help find the T problem's solution easily?

This paper is organized as follows: Section 8.3 is the background that provides an overview of some previous research. Section 9.4 describes the methods we used. Section 9.5 discusses the experiment. Section 9.6 is the discussion followed by the conclusion.

5.3 Background

In this section, we briefly review some relevant studies about Transfer Learning and system entropy.

5.3.1 Transfer Learning

Transfer Learning has been widely used in Machine Learning for knowledge transfer from one problem to another problem. Researchers have used Transfer Learning in a range of problem domains, including Natural Language Processing(NLP), classification, and other machine learning fields.

Jiang and Zhai [32] proposed a heuristic research method that implemented transfer learning in natural language processing (NLP). Their study of NLP focused on the following two factors:

- Different distribution instances of the source and target domains.
- Different conditional probabilities between the source and target domains.

They performed three sets of experiments as follows:

- Remove the misleading source domain instances.
- Add more weights to the target data or instances.

- Raise the training set by adding the unlabeled target data with their predicted labels.

They tested their method using three different datasets, and the results showed an improvement in the target predictions. They concluded that the most effective technique to exploit information from the target domain is by using the target's instance weights.

Zadrozny [79] proposed a bias correction method. This method calculates the estimation probability of selected sample data and gets the correct distribution of the unbiased samples by rejected samples. From a transfer learning perspective, this method can be used to determine the differences in marginal probability distribution between the source and the target domains. The author used the KDD-98 dataset to run the experiment. The method can be used for learning data classifiers and classifier evaluations. He categorized the sample selection bias classifier learning categorized into two types:

- The local type, which depends on the data predictive function.
- The global type, which depends on the data predictive function and marginal distribution.

Jiayuan et al. [31] proposed an algorithm called kernel mean matching (KMM). This algorithm tries to learn the ratio between the source domain data and the target domain data. Then it recalculates the training points by calculating the kernel Hilbert space of the source and target means. One of the advantages of using KMM is that if the data set is small, the KMM avoids density estimation performance of conditional probabilities for either the source domain or the target domain. The authors tested this algorithm using regression and classification benchmarks from UCI. They claimed KMM showed great results, and in some cases, the KMM reweighting process outperformed reweighting using the true sample distribution.

5.3.2 Entropy

Entropy has been used to make performance improvements in evolutionary algorithms. S.K. Smit and A.E. Eiben [60], claimed entropy can discover relevant parameters of Evolutionary Algorithms(EA). They explained an algorithm called REVAC (stands for Relevance Estimation and Value Calibration) that can estimate EA parameters. The main contribution of their study was to improve the EA performance by tuning EA parameters. The results showed the advantage of using this direction.

Vargas et al. [70] studied the entropy within GA populations. They proposed a method that used entropy time series of winds and earthquakes to generate the GA's initial population.

Their proposed method named Genetic Algorithm Proposed (GAP), and it aimed to improve the performance of the GA. The result showed the performance of the proposed algorithm improved compared to standard GA. They also claimed there is a positive correlation between GA performance and the initial population entropy.

These results show that the implementation of entropy with EA has yielded positive results on a number of problems. Our goal is to provide a case study that demonstrates how using entropy with TL to leverage the GA process can improve our ability to solve problems. Our study constructed a possible method for TL algorithm designers to improve their tasks.

5.4 Method

We employ the population in an EA as a model for the knowledge that the algorithm has about solving a particular problem. We created a model of TL implemented with GA. Our model starts with a random population to solve the S problem. After the S solver found the optimal solution to the S problem, we computed H and transferred parts of final population of S to the T problem solver. The GA solver solves the T problem using the transferred population. Then after the GA solver found the solution to the T problem we measured the H of the T final population. We plotted the H of both final populations of the S and T to compare and illustrate the differences (see Figure 5.1). However, our proposed algorithm generates the transferred population based on the H information of the source population.

Our experiments focuses on both the source and the target problem. We investigate how can we use information from the source problem and transfer it to the target problem. In these experiments a tunable fitness function is used to test our method on problems of varying difficulty. The fitness function we implemented is a type of recompense or reward function that must learn.

Experiment 1 starts by initialing a random population and passes to the $GA(S)$ solver to find the solution to the source problem S (See diagram Figure 5.1). The entropy of the source final population is measured and plotted. The source final population transfers to the target problem. The $GA(T)$ solver solves the target problem. The T final population entropy is measured and plotted. Entropy is computed as a measure of randomness in each bit position of the members of the population (see Equations 6.3 and 7.2a). The hope of measuring and plotting the entropy of the S and T populations is to show that the entropy values of each bit can measure and point the differences between the populations. This is a direct measure of

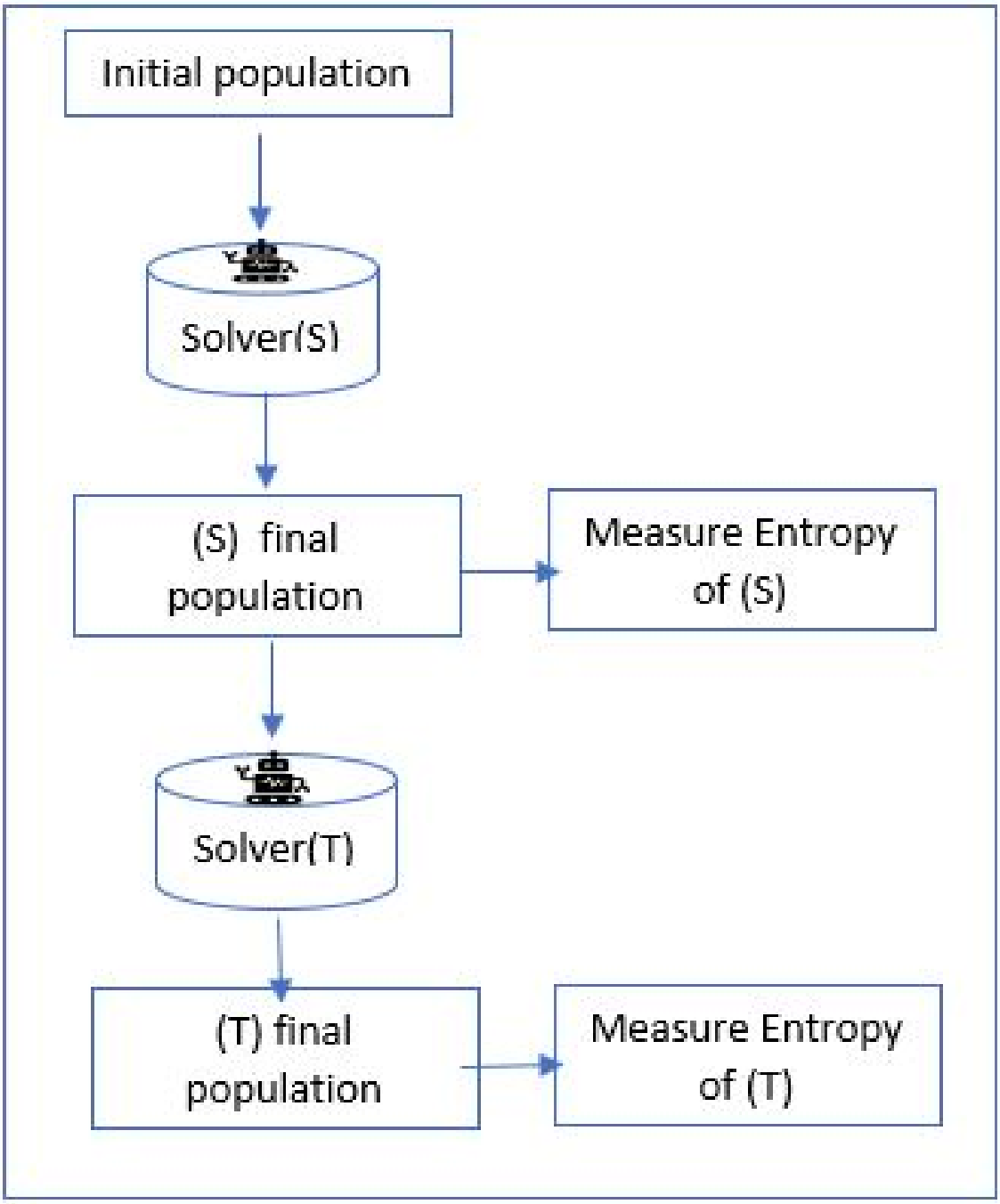

Figure 5.1: A random initial population is created and passed to the $GA(S)$ solver to solve the S problem. The source final population entropy is measured. Then, the final population of S transfers to the $GA(T)$ solver to solve the T problem.The T solver must find the optimal solution to the T problem. Then, entropy is measured the final population after T is solved.

what Estimation of Distribution Algorithms attempt to exploit.

The second experiment will attempt to exploit entropy information to determine what information should be transferred between the solver for S and T. We will explore transferring bits with low entropy from the source population to the initial population of the target problem.

The first experiment identifies bits that have knowledge useful for solving the S problem and bits that do not. The second experiment is designed to show how to efficiently move knowledge to the target problem from the source problem (for more detail see Section 9.5).

Genetic Algorithm (GA) is a search algorithm inspired by the reproduction and survival of individuals. A set of individuals is combined to make a population. GA is based on the generate-and-test style of algorithms. The candidates of each generation are tested through competition and the most successful individuals are selected to produce new candidate solutions. The competition is based fitness values. To generate new candidates, GA applies the following operations:

- Crossover operation is a binary mixing operation applied to two candidates producing one new candidate.
- The mutation is a unary operation that creates variation in the new candidates produced [46].

In our study, each individual or proposed solution consists of 40 bits, and the whole population is 100 individuals. Table 8.1 presents the full set of GA parameters used in our experiments.

Table 5.1: Genetic Algorithm Parameters

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	100
Mutation rate (per bit)	0.1
Crossover rate	0.05
Type of crossover	Uniform crossover
Tournament Size	5

Fitness Function is a measure of the quality or desirability of the an individual. To find the maximal fitness is to find the "best" answer. Our function is composed of sub-functions whose values are based on a subset of the bits. A similar idea was presented Heckendorn's paper on Embedded Functions [29]. The model used in this paper is a form of landscape problem, specifically the fitness function is defined as: $f : \mathcal{B}^n \rightarrow \mathbb{R}$ as:

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1]) \tag{5.1}$$

where s is the number of bits in the domain of subfunction g, m the number of subsections of s, and n is the total number of bits ($n = m * s$). $a_i \in \mathbb{R}$, functions $g_i : \mathcal{B}^s \to \mathbb{R}$, $x \in \mathcal{B}^n$, and the notation $x[a, b]$ extracts bits in positions a through b from bit string x.

In our experiments, each individual of the population consists of 10 sub-functions, g_i. Since our f is additively separable, the GA solver must find the optimal solution by solving some or all of the sub-functions of each individual. Our goal was to make each sub-function challenging to solve, so a **Deceptive Function** [26] was chosen for each of the sub-functions (g_i). Each of these functions is defined over s bits. It is deceptive in that if the bit count (the number of bits set to 1) of the argument b, denoted $bc(b)$, is s the maximum value is achieved. Otherwise the value is the number of zeros in b. This creates a deceptive function with b_{max} equal to all ones.

$$g(b) = \begin{cases} s & \mathrm{bc}(b) = s \\ s - 1 - \mathrm{bc}(b) & \text{otherwise} \end{cases} \tag{5.2}$$

We measured and plotted the entropy of the source and target populations (see Section 9.5). **Entropy** is measure from information theory associated with the predictability of random variable. As the value of entropy approaches 0 the random variable becomes a fixed constant. As the value of entropy approaches 1 the random variable become completely random. Entropy of a random variable X is denoted by $H(X)$ [58]. The entropy is calculated as follows:

$$H(X) = -\sum_{i=0}^{n} P(x_i) \log_2 P(x_i) \tag{5.3}$$

For a discrete random variable X with n possible values, x_i is the i^{th} possible value and $P(x_i)$ is the probability of x_i occurring in X.

In our study, $n = 2$ since we will be computing the entropy of the bits in a given bit position in the population and a bit can only be 0 or 1. The probability of 0 or 1 is determined by considering the all the bit values in a fixed position across the whole population.

We have two hypotheses that are linked to the two questions in Section 9.2. The first is that entropy will indicate which part or parts of the population have the most information for solving a problem. The second is that constructing the transferred population based on the low entropy values will preserve the identified knowledge from the source problem, and this will

result in speeding up the process of finding the optimal solution to a similar target problem. Some caveats with respect to the idea of "similar".

5.5 Experiment

In this section, we provided two experiments. The first experiment dealt with measuring and plotting the entropy of the source and target final population. The second experiment dealt with sampling the transferred population using the entropy and the source problem's final population.

5.5.1 First Experiment

For this experiment, we solved two different problems and measured the entropy of the final population of the source solution and the final population of the target solution. The GA solved the source problem and transferred that final population to initialize the population for solving the target problem. The entropy of **each bit** position was calculated using the following function:

$$H(X) = -P(x_i = 0)\log_2 P(x_i = 0) + -P(x_i = 1)\log_2 P(x_i = 1) \quad (5.4)$$

where the source and the target population each have 100 individuals. Each individual in consists of 40 bits. The entropy of each bit position, $H(X)$, was computed for a population.

5.5.1.1 First problem

Using the fitness function from Equation 7.2a, we chose the following as the source problem and the target problem:

$\vec{a_s} = (1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0)$

$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0)$

As we can see, the target fitness function is different than the source fitness function in one subfunction.

Figure 5.2 shows 40 bits and their entropies. These bits represent the final population of the source problem. This population contains the optimal solution to the source task. The curve of the plot shows low and high entropies. We want to draw the focus to the marked area between bits 20 and 23, the area surrounded by two dash lines. We will compare these bits to the entropies in the population for the target problem.

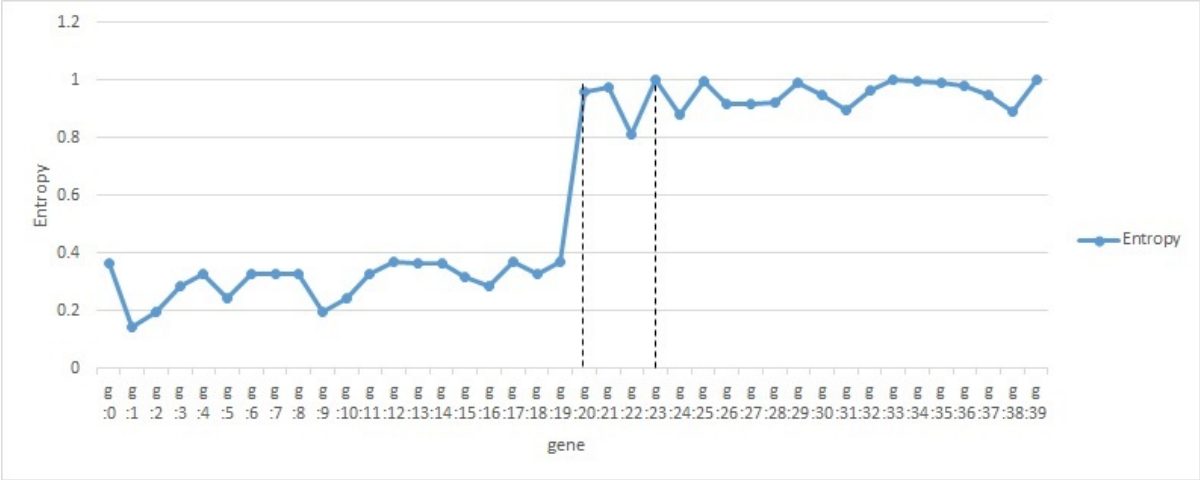

Figure 5.2: This is an example of the entropy of each bit position the GA solver found when solving the source problem.

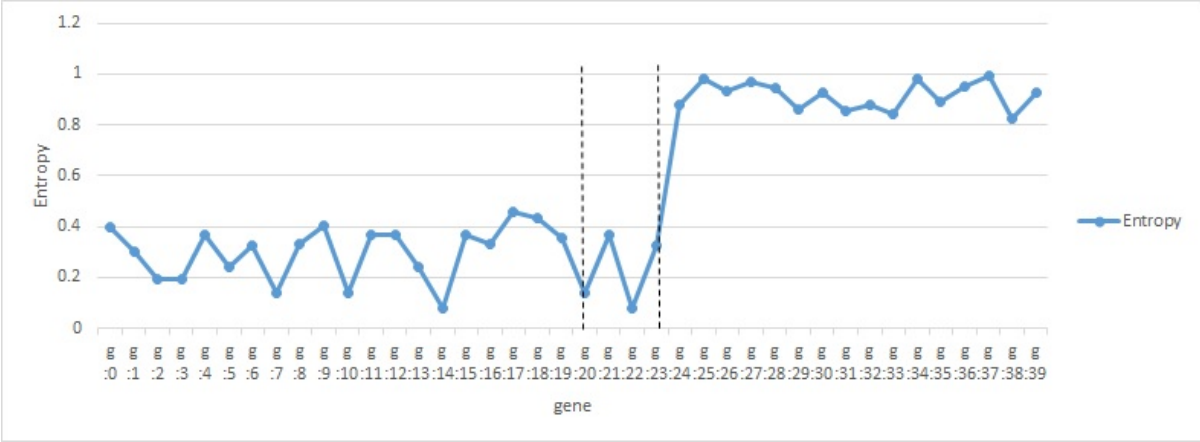

Figure 5.3: This is an example of the entropy of each bit position the GA solver found when solving the target problem.

Figure 5.3 shows 40 bits and their entropies. These bits represent the final population of the target problem. This population contains an optimal solution for the target task. Also, the curve of the plot here shows low and high entropies, but the low area is wider than the low area of the source plot in Figure 5.2. In Figure 5.3 bits (20-23) show a wider low curve than the source plot in Figure 5.2.

The area between the two dashed lines shows the difference between Figure 5.2 and Figure 5.3. The source solver does not care about including these sub-functions in its solution for bits (20 - 23). This is why these bits have high entropy. On the other hand, the target solver must add this knowledge to find the optimal solution. These sub-functions must be included in the solution of target problem.

The entropy can locate the needed additional knowledge. By looking at the target problem bits (20 - 23) in Figure 5.3, they now have low entropy. Also, the GA has found the optimal solution to T. They now have knowledge about how to solve the T problem, but in the source population in Figure 5.2, they do not have that knowledge.

5.5.1.2 Second Problem

In this experiment the target solver must add two new subfunctions of knowledge to solve the target problem.

$\vec{a_s} = (1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0)$

$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0)$

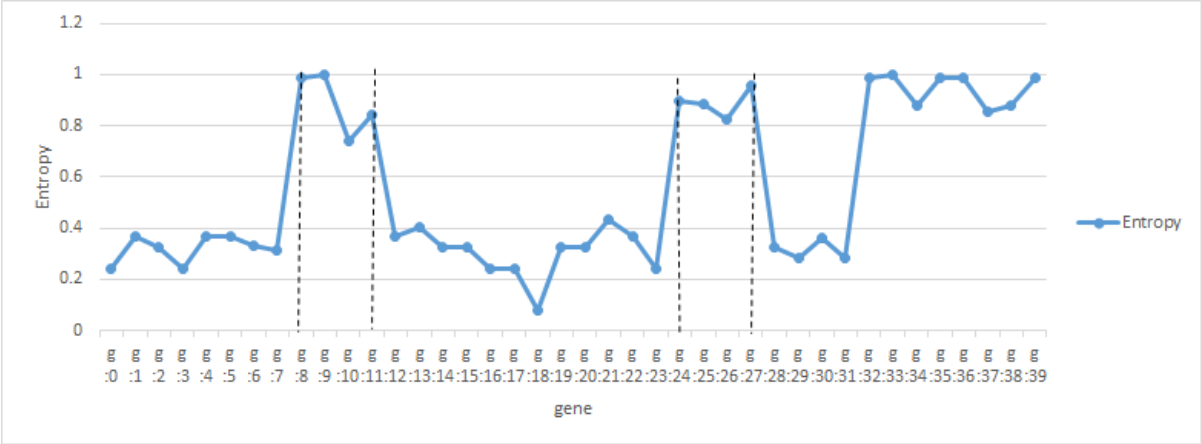

Figure 5.4: This an example of what the bits look like after the GA solver finds the solution to the source problem for the second problem. This plot shows 40 bits of the source population corresponding with their entropies. The plot shows bits with low entropies versus bits with high entropies.

Figure 5.4 shows 40 bits and their entropies. These bits represent the source's final population. This population contains an optimal solution to the S problem. There are two areas in the middle of the plot bits(8 - 11 and 24 - 27) that have high entropies of interest and we surround them with two dash lines.

Figure 5.5 shows 40 bits and their entropies. These bits are representing the target's final population. This population has an optimal solution regarding the T problem. We surround two areas of the plot bits(8 - 11 and bits 24 - 27) where they show low entropies compared to Figure 5.4.

Figure 5.4 and Figure 5.5 show two areas surrounded by two dashed lines. in Figure 5.4 it is clear that the source task did not include these two areas in its knowledge to find the source optimal solution; on the other hand, the target task includes these two areas in its knowledge

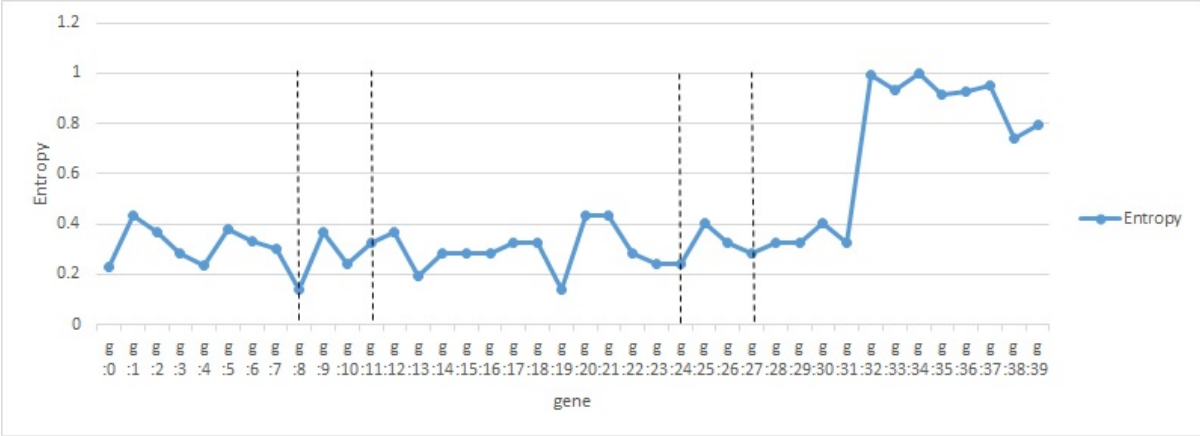

Figure 5.5: This is an example of what the bits look like after the GA solver finds the solution to the target problem. The target solver must add two subfunctions of knowledge to solve the target problem. This plot shows 40 bits of the target problem corresponding with their entropies. The plot shows bits with low entropies and bits with high entropies.

to find the target optimal solution, as seen in Figure 5.5. This is why we see these two areas with low entropy in Figure 5.5.

We can say now these two areas bits (8 - 11 and 24 - 27) have knowledge about solving the T problem, but before these two areas did not have any knowledge about the solution. The last part of the curve in all previous plots shows bits with high entropies (bits 33 - 39). These bits were not included in the solution of any problem S or T.

5.5.1.3 Discussion

From the first experiment, we can see how the bits of populations are represented along with their entropy. This shows that the bits have a range of entropy values. We can think about low entropy bits as the bits that already have a decision regarding the correct solution problem, i.e. these bits have converged on what the algorithm found to be the most promising solution. Regarding problems S and T in general, the bits that participate in the solution have low entropy value, i.e. these bits presumably have more information about solving the problem. Thus, transferring these bits will save time and effort for the GA on the second problem, if they share the many of the same subfunctions. Because these bits have already reached a decision for problem S and assuming that T is very similar in form, the second phase the algorithm can focus on how to solve the parts of problem T that are different.

Entropy Focus Point (EFP) is an value of estimated entropy that separates low values from high values. It is the point that distinguishes which bits have made decision from which bits have not. Also, this point distinguishes which bit of an individual has participated in the

solution of a problem from other individual bits. In this study, it seems that 0.45 makes a good EFP (see Figure 5.2, 5.3, 5.4 and 5.5). We use 0.45 as the EFP point in the Second Experiment 5.5.2.

5.5.2 Second Experiment

In this experiment, we sampled the solution population from problem S four different ways. We passed each one of them to the GA solver to find the optimal solution to the T problem. We counted how many generations the GA solver took to find the optimal solution. Each one of the target problems was solved 50 times. The following is a description of each type of the sampled population:

- **GA** : a copy of the initial population that passed to the source solver. This copy was passed to the target GA solver as a control.
- **EntropySampled** : a sampled population that uses our proposed algorithm below.
- **ED** : the Estimation of Distribution sampling strategy. This uses the probability of a 0 or 1 occurring in each bit position to generate an initial population for the target GA solver. (For more information see [36])
- **TL** : an identical copy of the source final population. That is, just start from where the source GA solver left off.

In **our proposed algorithm**, we used the bit entropy of the final source population and the knowledge of the source final population to create the transfer population. The following steps summarize our proposed algorithm for generating a transfer population between the S solver and the T solver.

1. Measure the entropy of the source final population (see Section 9.4).
2. Transfer best individual's bits of the source population whose entropy less than or equal to EFP, which is equal to 0.45.
3. Generate random bits for bit positions with high entropy.

In this experiment, we chose one source problem and three different target problems. We began by generating an initial random population and solving the source problem. We made an

identical copy of the initial population and passed it to the target solver as the control (GA). Then we experimented with the Transferred Learning approach. We constructed the transfer population from three different sampled populations (EntopySample, ED, and TL). We did the same steps for each target problem where we counted how many generations the target solver took to find the optimal solution for each strategy. This may give us a better understanding of how knowledge is transferred and what to transfer.

The following is the source problem using the formula from Equation 7.2a:

$$\vec{a_s} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)$$

The following are the three target problems: 7.2ca, 7.2cb, and 7.2cc, the first problem adds one subfunction of knowledge to the target problem. The second problem adds three new subfunctions of knowledge to the target problem. The third problem adds seven new subfunctions of knowledge to the target problem and can be considered to not be similar to the source problem.

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{5.5a}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0) \tag{5.5b}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{5.5c}$$

5.5.2.1 First Target Problem

The first target problem (Equation 7.2ca) requires the GA solver to add one new subfunction of knowledge to the source population to solve the target problem. Figure 5.6 describes how many generations the GA solver took to find the optimal solution to the first target problem.

Figure 5.6 represents the result of the first target problem. It shows a boxplot of the test generations for each strategy we implemented. As can be seen, our algorithm strategy of population sampling (EntropySampled) has the lowest number of generations and it exceeds all other strategies. The GA solver must add one new subfunction of knowledge to solve the target problem.

We ran the Wilcoxon rank sum test between the GA and EntropySample strategies and the p-value was equal to 3.0312. That is enough statistical evidence to show that sampling the transferred population using the proposed algorithm has successful improvement. We also used

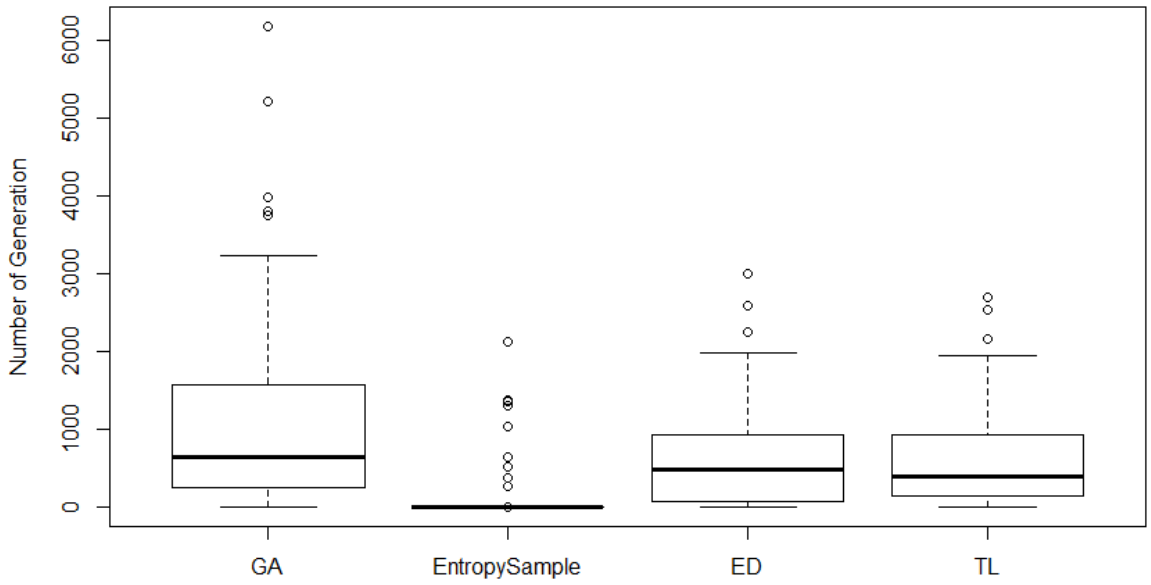

Figure 5.6: The figure shows a quartile or a boxplot diagram that shows the distribution of the number of generations the GA solver took to find the optimal solution of the first target problem based on 50 replicates. The GA solver must add one new subfunction of knowledge to solve this problem. The EntropySampled population shows low generation numbers compared to the other three populations.

the pairwise Mann-Whitney U test, which is a statistical comparison test of the three sampled strategies of the transferred population(EntropySampled, ED, TL), as shown in Table5.2.

Table 5.2: The pairwise Mann-Whitney U test for target problem one, to indicate if the differences of each strategy are significant or not. Each cell of the table has a p-value of this test. For our study, the p-value of this test is < 0.05, which indicates that the median value of the generation number is significant when compared to the other strategies. The bolded values of the table show where the significant improvement is.

	ED	EntropySample	GA
EntropySample	**1.509**	-	-
GA	**0.031**	**3.012**	-
TL	0.882	**1.208**	**0.049**

5.5.2.2 Second Target Problem

The second target problem (Equation 7.2cb) requires the GA solver to add three new subfunctions of knowledge to the source population to solve the target problem. Figure 5.7 shows how many generations the GA solver took to find the optimal solution of the second target problem.

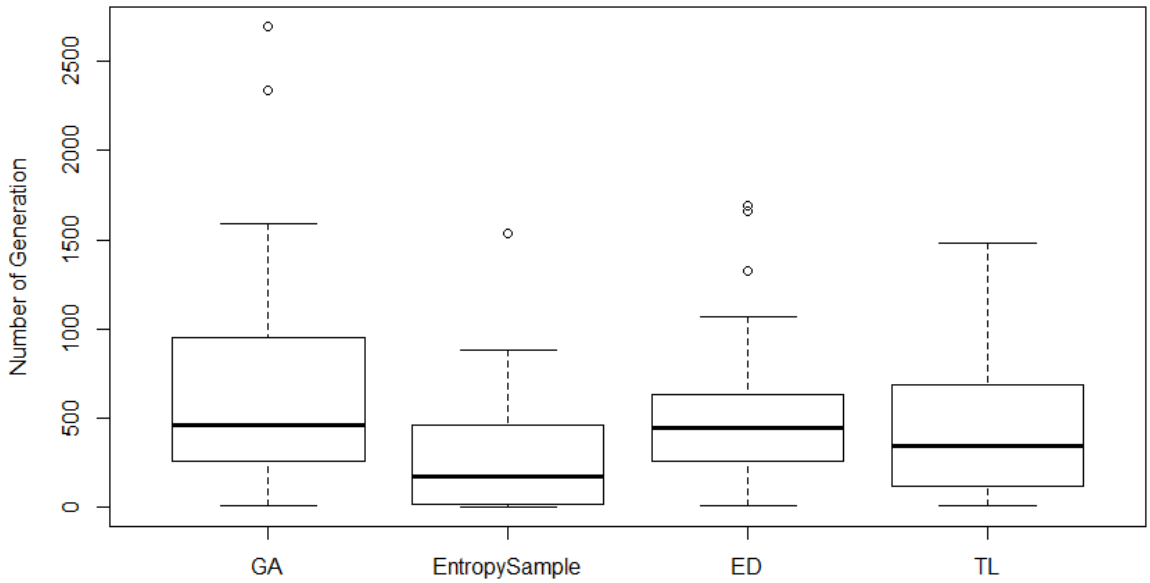

Figure 5.7: The figure is a quartile or a boxplot diagram that shows the distribution of the number of generations the GA solver took to find the optimal solution of the second target problem based on 50 replicates. The GA solver must add three new subfunctions knowledge to solve this problem. The EntropySampled population shows low generation numbers compared to the other three populations.

Figure 5.7 represents the result of the second target problem. The GA solver must add three new subfunctions of knowledge to solve the target problem. It shows a boxplot of the test generations for each strategy we implemented. As can be seen, our algorithm strategy of population sampling (EntropySampled) has the lowest number of generations and it exceeds all other strategies.

We ran the Wilcoxon rank sum test between the GA and EntropySample, strategies and the p-value was equal to 0001. That is enough statistical evidence to show that sampling the transferred population using the proposed algorithm has successful improvement. We also used the pairwise Mann-Whitney U test, which is a statistical comparison test of the three sampled strategies of the transferred population(EntropySampled, ED, TL), as shown in Table5.3.

5.5.2.3 Third Target Problem

The third target problem (Equation 7.2cc) requires the GA solver to add seven new subfunctions of knowledge to the source population to solve the target problem. Figure 5.8 describes how

Table 5.3: This is the Pairwise Mann-Whitney U test for target problem two, to indicate if the differences in the number of generations of each strategy are significant or not. Each cell of the table has a p-value of this test. For our study, the p-value of this test is < 0.05, it indicates that the median value of the generation number is more significant than others strategy. The bolded values of the table show where the significant improvement is.

	ED	EntropySample	GA
EntropySample	**0.001**	-	-
GA	0.452	**0.000**	-
TL	0.317	**0.019**	0.070

many generations the GA solver took to find the optimal solution to the third target problem.

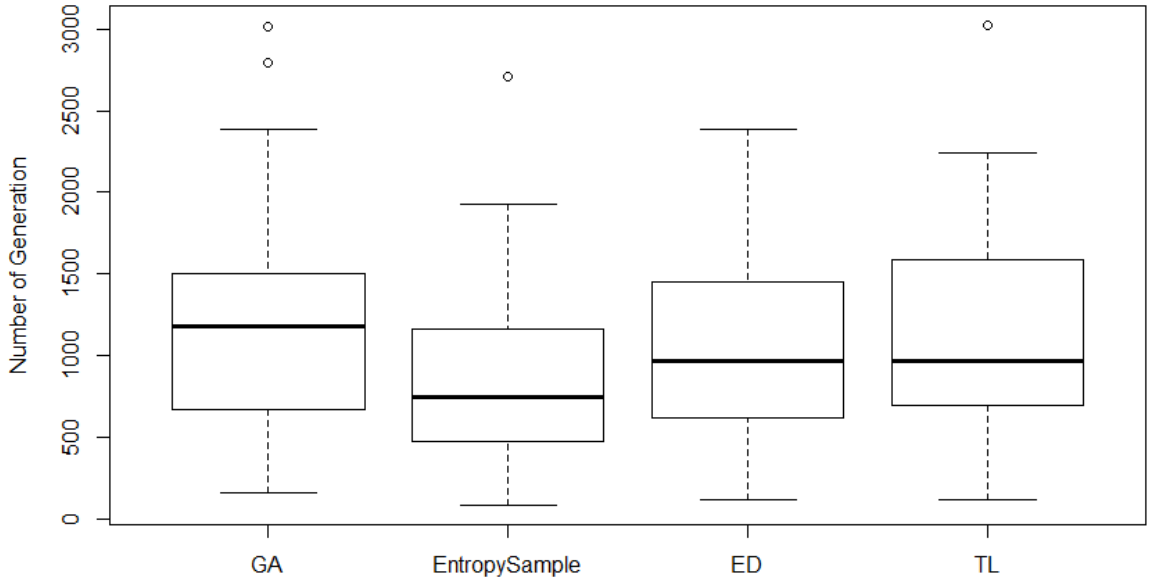

Figure 5.8: The figure is a quartile or a boxplot diagram that shows the distribution of the number of generations the GA solver took to find the optimal solution of the third target problem based on 50 replicates. The GA solver must add seven new knowledge to solve this problem. The EntropySampled population shows low generation numbers compared to the other three populations.

Figure 5.8 represents the result of the third target problem. It shows a boxplot of the test generations for each strategy we implemented. As can be seen, our algorithm strategy of population sampling (EntropySampled) has the lowest number of generations and it exceeds all other strategies. The GA solver must add seven new subfunctions of knowledge to solve the target problem.

We ran the Wilcoxon rank sum test between the GA and EntropySample strategies and the p-value was equal to 0.002. That is enough statistical evidence of sampling of the transferred population using the proposed algorithm to show successful improvement. We also used the pairwise Mann-Whitney U test, which is a statistical comparison test of the three sampled strategies of the transferred population(EntropySampled, ED, TL), as shown in Table 5.4.

Table 5.4: This is the Pairwise Mann-Whitney U test for target problem three, to indicate if the differences of each strategy are significant or not. Each cell of the table has a p-value of this test. For our study, the p-value of this test is < 0.05, which indicates that the median value of the generation number is more significant than the other strategies. The bolded values of the table show where the significant improvement is.

	ED	EntropySample	GA
EntropySample	**0.032**	-	-
GA	0.335	**0.002**	-
TL	0.609	**0.006**	0.676

5.6 Discussion

The success of the GA solver depends on the content of the transferred population. As we can see from Tables and Figures, the number of generations the GA solver took to find the optimal solution is differentiated from one strategy to another. How we sampled the transferred population matters even though each of our strategies, except the control, was sampled from the same source population.

Entropy gives us an estimation of how much each bit knows about solving the problem, instead of, starting from scratch or sampling raw data. The entropy information helps us to discover the bits from the solution of S that have knowledge of solving problem T.

Our proposed algorithm sampled the transferred population based on the entropy information of the source final population. From the graphs and tables, we can see how much our algorithm improves over the other strategies tried (ED and TL).

Future work: The Entropy Focus Point (EFP) is a point that distinguishes between low entropy values and high entropy values. This point helped us to find which part of the source population has a piece of information about solving the S problem. For this study, we estimated the EFP by examining plots of entropy and those plots illustrated what these values look like. Clearly we would like to algorithmically ground this approach and support our choice empirically. We are trying several approaches as well as other ways to compose initial populations for the $GA(T)$ problem solver.

5.7 Conclusion

Transfer Learning helps ML designers to speed up the process of finding optimal solutions. Learning from the solution of similar problems is not only natural, but makes computational sense. Transferring knowledge to the target solver has an important impact in reducing the time and effort of finding the optimal solution to the T problem. Finding what information to transfer to the target solver improves the solver process.

In order to answer the above two questions, we ran two experiments. The first experiment dealt with computing the bit's entropy of the S and T final population and plotting them. The second experiment tested our proposed algorithm and how to improve the quality of transferred information from the source population to the target solver.

One of the exciting advantages of transfer learning and our proposed algorithm is that it addresses the cold start problem. The GA solver does not have to start from scratch to solve the target problem if it is within a domain of solved problems. It attempts to leverage knowledge of the source problem to solve the new problem.

5.8 Acknowledgments

This research is supported by Saudi Arabian Cultural Mission (SACM) and Al Baha University, Al Baha, Saudi Arabia. I would like to give very special thanks to my supervisor Dr. Robert B. Heckendorn for his great advice in this research.

Chapter 6

Quantifying the Right Combination of Knowledge and Population Diversity for Transferred Populations

This chapter is exactly as accepted at the 2021 International Conference on Computational Science Computational Intelligence conference (CSCI-2021). It addressed the third objective, which is "Quantifying the Right Combination of Knowledge and Population Diversity for Transferred Populations". This chapter studies how to quantify the right combination of the previous knowledge and population diversity of the transferred population. The results of evaluating this study showed that the right combination of the transferred population improves the performance of the GA. This paper answered the following tasks:

Q.1 : Study the right quantities of each combination of the transferred population (old knowledge and population diversity).

Q.2 : Investigate what the other factors are that may affect this combination.

6.1 Abstract

Transfer Learning (TL) is a process of leveraging knowledge from one problem called the source problem (S) to a related problem called the target problem (T). Experiments and previous research (see Section 8.3) showed that the knowledge from solving a related problem and population diversity are able to aid machine learning algorithm to solve a related problem more easily. To understand the right amount of knowledge and diversity of the transferred population, we modeled the TL process using Evolutionary Computation (EC). Our model gives us more access and control over the TL components, especially the content of the transferred

population. The results showed that the relationship R between the source S and the target T problems can quantify the right combination of each component of the transferred population.

6.2 Introduction

Transfer Learning (TL) is a process of leveraging knowledge from one problem called the source problem (S) to a related problem called the target problem (T). TL aims to transfer information from the S problem to the T problem. TL has shown promising results; for example, TL can reduce the time and effort of finding a solution to a new related problem. In cases where there is not enough training data, TL can combine data from multiple sources [53, 69].

The S and the T relationships can also indicate if the problem is more or less difficult. For example, if someone is accustomed to playing ping-pong, it is very common for them readily adapt to playing tennis because, ping-pong and tennis are types of racket sports that use a racket and a ball. In the TL environment if the S and the T problems are related to each other, the process of TL is more reliable and has more accurate results. If the S and T problems are not related to each other, we may face a negative transfer situation.

As we discussed above, the TL has information about solving the S problem and it aims to transfer this information to the T problem. According to a study performed by Gupta and Ong [28], TL depends on old knowledge and population diversity. The question, then, is what is the right combination of the previous knowledge and diversity. Answering this question will help TL designers to implement TL in Machine Learning (ML) problems more successfully. The aim of this study is to understand the right combination of old knowledge and population diversity. We also aim to determine if there are other situations that may influence this combination. One benefit of discovering the accurate combination is to minimize the effect of negative transfer.

To achieve our aim, we modeled the process of TL using Genetic Algorithms (GA) [21] and system diversity [58]. System diversity can be thought of as entropy (H) from the information theory concept. The process of our model is as follows: The GA will enable our model to generate solutions of the S and T problems and will find the optimal candidates among all solutions. TL will transfer the knowledge between the S and the T problems. System entropy will measure the diversity of the transferred population. We will count how many generations the GA solver took to find the optimal solution for each case (see Section 8.4). By measuring the number of individuals transferred from the S problem and the population diversity of the transferred population, we can determine the right combination for the transferred population.

The purpose of this study is to answer the following questions:

- **Q1** What are the right proportions of each combination of the transferred population (old knowledge and population diversity)?
- **Q2** What are other factors that may affect this combination?

Researcher has shown that, the transferred population must contain knowledge of solving the old problem and population diversity. Gupta and Ong [28] concluded that knowledge of the related problem and population diversity are two sides of the same coin, and those two components are the ingredients of a successfully transferred population.

Our results showed that the relationship R between S and T can quantify the amount of each component of the transferred population. The R between the S and the T can quantify which component of the transferred population the TL designers must use when they create the transferred population. If the relationship between the S and the T is close, the knowledge of the old problem is more important than the population diversity. On the other hand, if the relationship between the S and the T is large that is the problems are not related to each other, the diversity is more important than the knowledge. But this does not mean one of the two-components can be completely absent. (see Section 9.6 for more details).

This paper is organized as follows: Section 8.3 provides the background for the study and summarizes relevant research; Section 8.4 outlines the method; Section 9.5 explains the experiment; Section 9.6 clarifies the discussion followed by the conclusion and my acknowledgment.

6.3 Background

This section briefly discusses some TL researchers, system diversity, and Genetic Algorithms (GA).

This section briefly discusses some TL researchers, system diversity, and Genetic Algorithms (GA).

6.3.1 Transfer Learning

Transfer Learning strategies have been implemented in many Machine Learning (ML) algorithms. Most of these researches used the whole concept of TL; for example, they transfer the knowledge of one problem to aid the solution in a related problem. TL has been implemented in the word classification field [37], networks [44], and dimensionality reduction [50].

Li et al. [37] proposed a TL method used in sentiment and topic lexicon co-extraction. They claimed their method generated accurate information to construct lexical information. They utilized the relation between selected words and the topic of the source domain to label the data in the target domain. The method they proposed has two stages. The first stage involves establishing a relation between the source and the target domains by identifying the sentiment words or seeds. After the sentiment words are identified, they generate a topic for these words in the target domain by mining some relation between these words and topic words from the source domain. The second stage of their work entailed proposing a new algorithm called Relational Adaptive bootstraPping (RAP). The aim of this step is to develop the words in the target domain. Their work provided a study of cross-domain sentiment analysis, a method that has two steps of transfer learning, and they evaluated their work extensively and showed great results.

Lilyana et al. [44] have studied Transfer Learning with Markov logic networks (MLNs). MLN is a probabilistic model that applies the idea of Markov in first-order logic. The authors implemented a transfer learning system called TAMAR. TAMAR performed its task of transferring the knowledge in two steps:

- Map the structure (MLN) of the source to the target domain.
- Correct the (MLN) weights of the source domain to be improved in the target domain.

They tested this model using real-life datasets like IMDB, UW-CSE, and WebKB. The results showed that this model reduced the time and amount of training data that are needed to train the model.

Sinno et al. [50] proposed a transfer learning algorithm based on dimensionality reduction. They also proposed a new feature reduction method called Maximum Mean Discrepancy Embedding (MMDE). Their algorithm consists of two steps:

- Learn the low dimensionality of both domains by applying MMDE.
- Use low dimensionality data of the MMED (from the first step) to train the data.

They ran two different experiments: one used WiFi data and the other one using text data classification of Reuters-21578. The results showed that their method performed well, and MMDE can be used for dimensionality reduction, regression, and classification for transfer learning.

6.3.2 System Diversity

Population diversification is an important factor for measuring the solution space of a problem. Population diversity can be measured using the system entropy H [58].

Peng et al [54] proposed an algorithm called Active Transfer Learning (ATL). The aim of this algorithm is to minimize the effect of the negative transfer problem. the ATL algorithm looks for samples from the source problem that are most relevant to the target problem. The authors used system diversity to ensure the selected samples of the source problem are the samples most relevant to the target problem. They evaluated the ATL algorithm using five different data sets. The results showed the ATL algorithm is more effective compared to other TL methods.

Rufer and Plank [57] performed a study on TL and Natural Language Processing (NLP). They were concerned with three tasks of NLP: sentiment analysis, tagging, and parsing. They proposed an approach to learn data selection using Bayesian Optimization. The authors mentioned that most of the old studies talked about the importance of the similarity between the TL problems (S and T). They proved the diversity is as important as the similarities of the TL problems. They evaluated their approach using different data sets like Amazon customer reviews. The results showed that this approach outperformed other approaches.

6.3.3 Genetic Algorithm

A **Genetic Algorithm** (GA) [21,46] is a type of Evolutionary Computation Algorithm [12,20] that is inspired by the process of natural selection. A GA is highly recommended type of algorithm for generating solutions to search and optimization problems. The process of a GA depends on mutation, crossover, and selection operators. In GA, selected candidates to a problem are called populations, and each candidate solution is called an individual. In atypical GA Each individual has a string of $\{0, 1\}$ representation of genes.

The evolution process usually starts by generating a random population and evolves using GA operators to find the optimal solution to the problem. Each evolving operation is called a generation. The fitness value of each individual is evaluated to find the optimal solution in each generation. An individual fitness value represents how successful this individual is or how close it is to the optimal solution. The fittest individuals of the current population are usually selected and altered by GA operations to generate next generation. Usually, this process is repeated until the optimal solution is generated or according to a fixed number of repetitions.

The GA requires the following:

- A gene representation of each individual.
- A fitness function to evaluate each individual.

Studies have shown that TL using GA improves the performance of other algorithms'. We aim to provide a case study that illustrates how TL can create a positive change on other algorithms performance, especially GAs.

6.4 Methods

The GA strategy was chosen to model the TL process. GA has the ability to characterize the final population of the S and T problems. Also, it can characterize the transferred population. The difficulty of the problems can be controlled by changing the fitness value and gene representation. The combination of the transferred population can be discovered by counting how many individuals are transferred from the S final population to the transferred population and measuring the diversity of the transferred population. These abilities allow us to study many hidden features of TL.

For this study, we used one static source problem S and three different target problems T. Our model transferred the knowledge from the S problem to the T problem. We chose this setting because it gives us control of most difficult situations. For example, we can measure the differences between S and T problems easily. The three different target problems vary between easy and hard cases. We will count how many generations the GA solver took to find the optimal solution. This way allows us to find what influences affect the transferred population in different situations.

Our model mixed the TL and GA together. The model started by generating many solutions to solve the S problem and looking among them to find the optimal solution. The TL transferred knowledge of the S problem to the T problem. The diversity of the transferred population was measured and plotted. Also, our model counted how many generations the GA(T) solver took to find the optimal solution of the T problem.

For this study, we used generational GA. The population consists of 100 individuals. Each individual consists of 40 bits. The tournament selection was implemented where the best n is chosen for population reproduction. In this study, the $n = 5$. For more details of the GA parameters see Table 8.1.

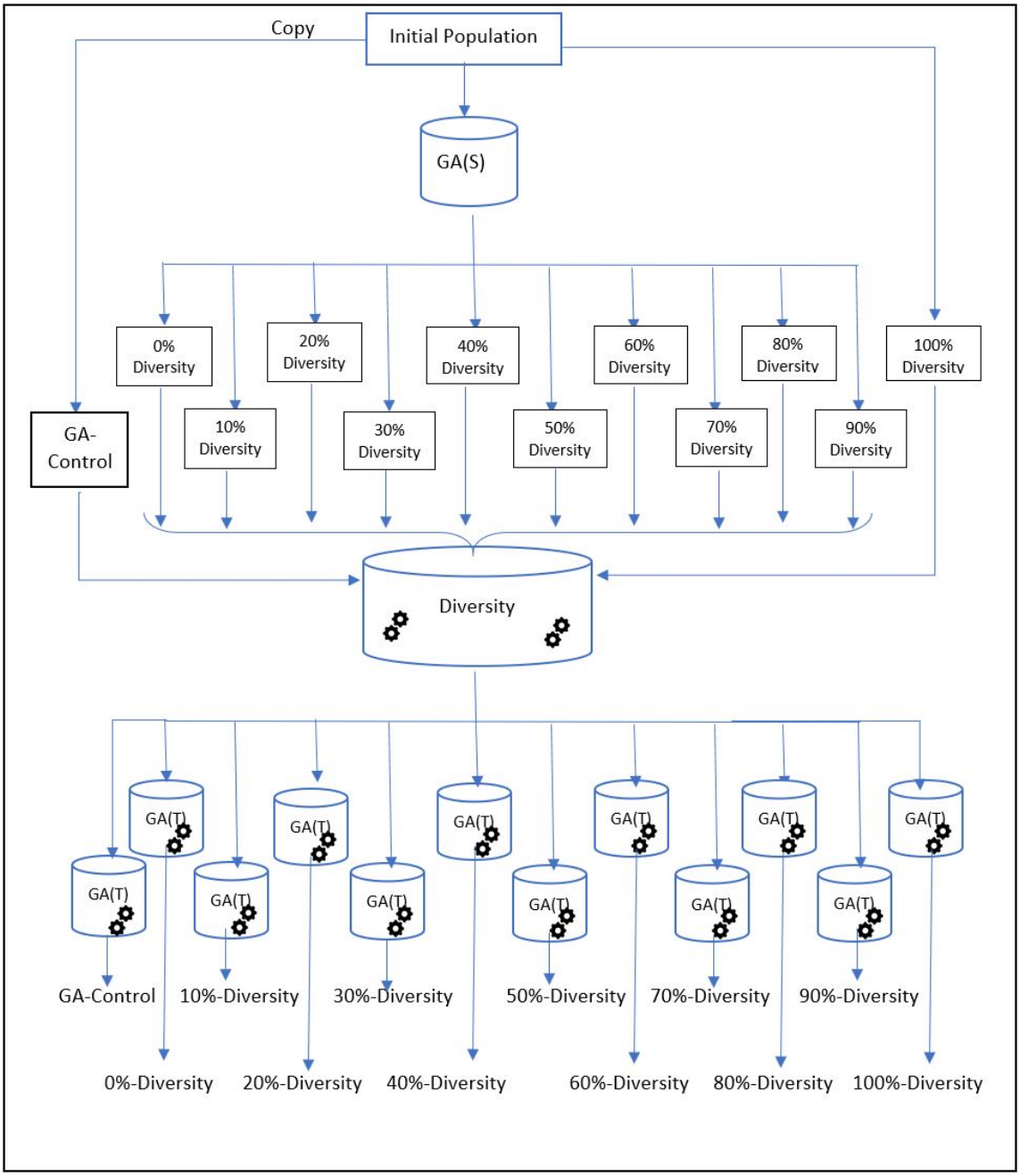

Figure 6.1: The $GA(S)$ starts solving the S problem by using a randomly initialized population. GA-Control is a copy of the initialized population that passed to the $GA(T)$ to find a solution for the T problem. $GA(S)$ solver solves the S problem. We then created 11 copies of the S final population. Each version is different than the other by 10% of the diversity amount. 0%-Diversity version of the transferred population is a copy of the best individual of the S problem. Then the amount of diversity is increased by 10% for each version until we have 100% diversity where the transferred population is a randomly initialized population. The diversity of all versions of the transferred population is measured. Then they passed to the $GA(T)$ to count how many generations the $GA(T)$ took to find the optimal solution of the T problem.

Table 6.1: Generational Parameters of the Genetic Algorithm

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	100
Mutation rate (per bit)	0.1
Crossover rate	0.01
Type of crossover	Uniform crossover
Tournament Size	5

Our experiment was designed to solve target problem where the target problem required the $GA(T)$ solver to add more elements of knowledge compared to the S problem to find the optimal solution. The fitness we implemented required the $GA(T)$ solver to learn new information in order to find the optimal solution. By changing the fitness value, we can discover different and harder cases.

The **fitness function** is a quality measurement of each individual. The GA will try to maximize the fitness values of each individual in order to find the best answer, which is the best fitness value of an individual. The fitness value represents how close each individual is to the optimal solution. Our fitness function collected many subfunctions. These subfunctions are defined by number of bits (see Equation 7.2a). We used a function called an Embedded fitness function that has been used by Heckendorn [29].

n bit space: $f : \mathcal{B}^n \rightarrow \mathbb{R}$:

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1]) \tag{6.1}$$

We define the fitness function over n bit strings, where g_i is the number of subfunctions. Gene x consists of m subfunctions; in our case, these m are not overlapped. $n = m * s$ which is the total bits string. In this study $m = 10$, $n = 40$, $s = 4$, and a_i $\{0, 1\}$. The s string forms the notation $x[a, b]$ it extracts a through b bits positions.

In this experiment, each individual consists of 10 subfunctions, g_i. The model subdivides the n string of bits into many subsections; for this study we have 10 subfunctions. Each subsection is 4 bits long. The **Deceptive Function** (DF) was chosen to solve each subfunction (see Equation 7.2b). It is a hard and misleading type of function. The GA must solve each subfunction to find the optimal solution. The GA applied the DF function to each 4-bit section and then multiplied by a corresponding weight. Function g_i is multiplied by a_i. If $a_i = 0$ the function is ignored. The a_i vector essentially indicates which sections of information in the n bit string

are important and which are not.

$$g(b) = \begin{cases} s & \text{bc}(b) = s \\ s - 1 - \text{bc}(b) & \text{otherwise} \end{cases} \tag{6.2}$$

For population diversity, we used entropy from information theory to measure the transferred population diversity. The system entropy is denoted as (H). It measures the randomness of a system by comparing symbols. Entropy values indicate if the next symbol is more predictable or not. If the entropy value approaches 0 or close to 0 then the next symbol is more predictable. If the entropy value is 1 or close to 1 then the next symbol is not predictable [58]. The following function (Equation 6.3) was used to calculate the diversity:

$$H(X) = -\sum_{i=1}^{||X||} P(x_i) \log_b P(x_i) \tag{6.3}$$

where X is a discrete random variable. The x_i is the i^{th} possible value of the gene. The $P(x_i)$ is the probability of x_i. The n is possible value and it can be either $\{0, 1\}$ in our study $n = 2$.

In this study, population diversity (Equation 6.4) computes the value of a bit in bit positions in the transferred population. As the following:

$$D(P) = \sum_{i=1}^{n} H(P[i]) \tag{6.4}$$

where each possible bit can be either 0 or 1. The bit probability $P(x_i)$ is determined by computing all bits in the same position across the whole population.

The study discussed in the previous section mentioned that old knowledge and population diversity are the main ingredients of the transfer population [28]. We hypothesize if the differences between the source problem S and the target problem T are small amounts, for example, one or two elements of knowledge, a small amount of diversity, for instance 10% to 40% of the transferred population is enough to lead the GA solver to find the optimal solution to the target problem. On the other hand, if the differences between the source problem S and the target problem T are big amount, e.g., eight or nine elements of knowledge, a huge amount of population diversity, e.g., 70% to 90% population diversity of the transferred population is required to help the $GA(T)$ solver to find the optimal solution.

6.5 Experiment

In this section, we describe our experiment. Our model started by solving the S problem using $GA(S)$ and randomly generated population. Then we created the transferred population. We created 11 different versions of the transferred population. Each version had a different amount of transferred population and diversity. For example, the first version had 100% copy of the best individual of the S final population as the transferred population and 0 total diversity; then each subsequent version varied in terms of transferred population and diversity by 10% until we had a transferred population with 0 amount of transferred knowledge and 100% diversity. We measured and plotted the diversity of these versions. Then, these versions of transferred populations passed to the $GA(T)$ solver to solve each one of the target problems. We counted how many generations the $GA(T)$ solver took to find the optimal solution.

The following is the source problem S:

$$\vec{a_s} = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0).$$

We have one static source problem and three different target problems. We aimed to transfer knowledge from the S problem to different T problems. This way, we could find out what other influences may affect the $GA(T)$ performance.

The following are the target problems:

$$\vec{a_t} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{6.5a}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0) \tag{6.5b}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{6.5c}$$

As we stated above, each one of the T problems (Equations 7.2c) is different from the S problems. The first T problem 7.2ca is different by one element of knowledge. The second T problem 7.2cb is different by four elements of knowledge. The third T problem 7.2cc is different from the S problem by eight elements of knowledge.

We computed the diversity of each version of the transferred population. We used entropy from the information theory (see Section 8.4 Equations 6.3 and 6.4). The following diagram (Figure 6.2) describes the population diversity of each version. Also, because we calculated the diversity of the transferred population from the same source population and before passing

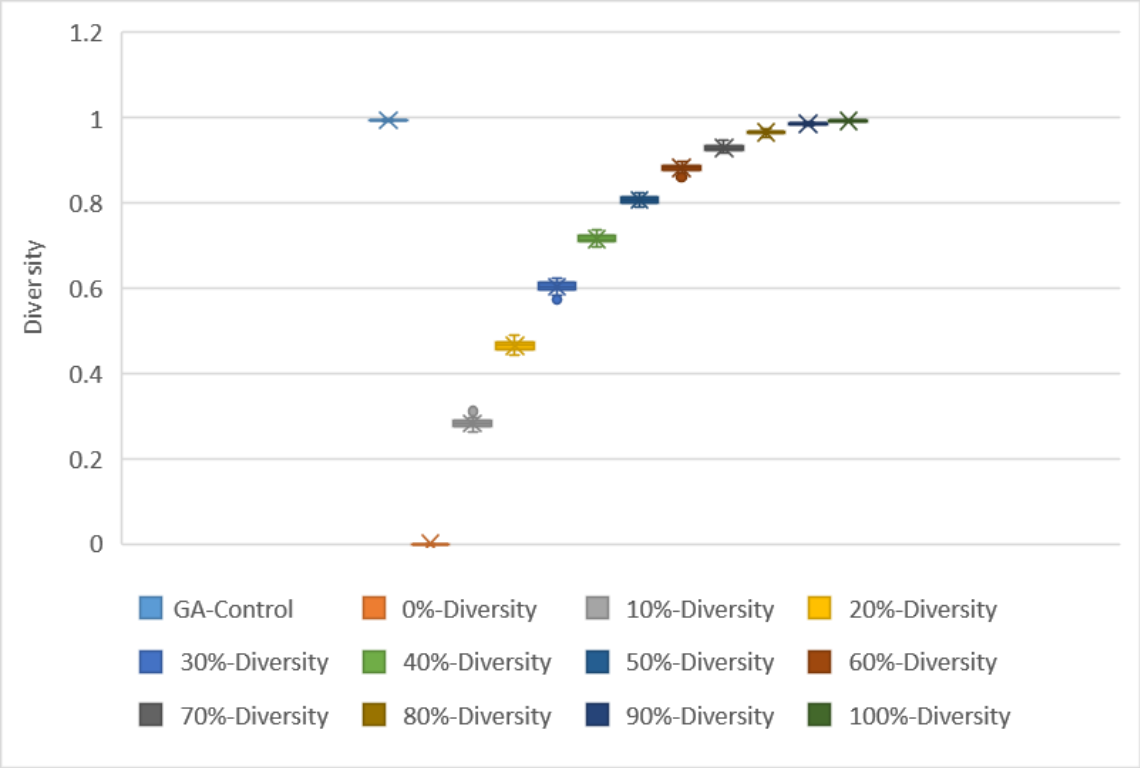

Figure 6.2: The diversity distribution diagram describes the amount of diversity of each version of the transferred population. Each version varies between 0 diversity until 100% diversity. The control unit population is a copy of the generated population that solved the source problem. 0%-Diversity is the less diverse population because it is just a copy of the best individual of the source population. The 100%-Diversity population is the most diverse population because it is totally random population.

them to the target solver $GA(T)$, we have the same diversity distribution for all three target problems (see Equations 7.2c).

6.5.1 First Target Problem

The first target problem 7.2ca is the problem in which the target solver must find the optimal solution of the T problem by adding one element of knowledge. We counted how many generations the $GA(T)$ took to find the optimal solution. The following graph (Figure 6.3) is a description of how many generations the GA(T) took to find the solution.

To make our study statistically sound, we chose to run the "Pairwise Mann Whitney U test" because the distribution of generations to the optimal solution is not normal. This type of test is a multiple comparisons test of the 11 versions of the transferred population we created, as shown in Table 6.2.

Table 6.2 shows the p-values of the 11 versions of the transferred population for solving the

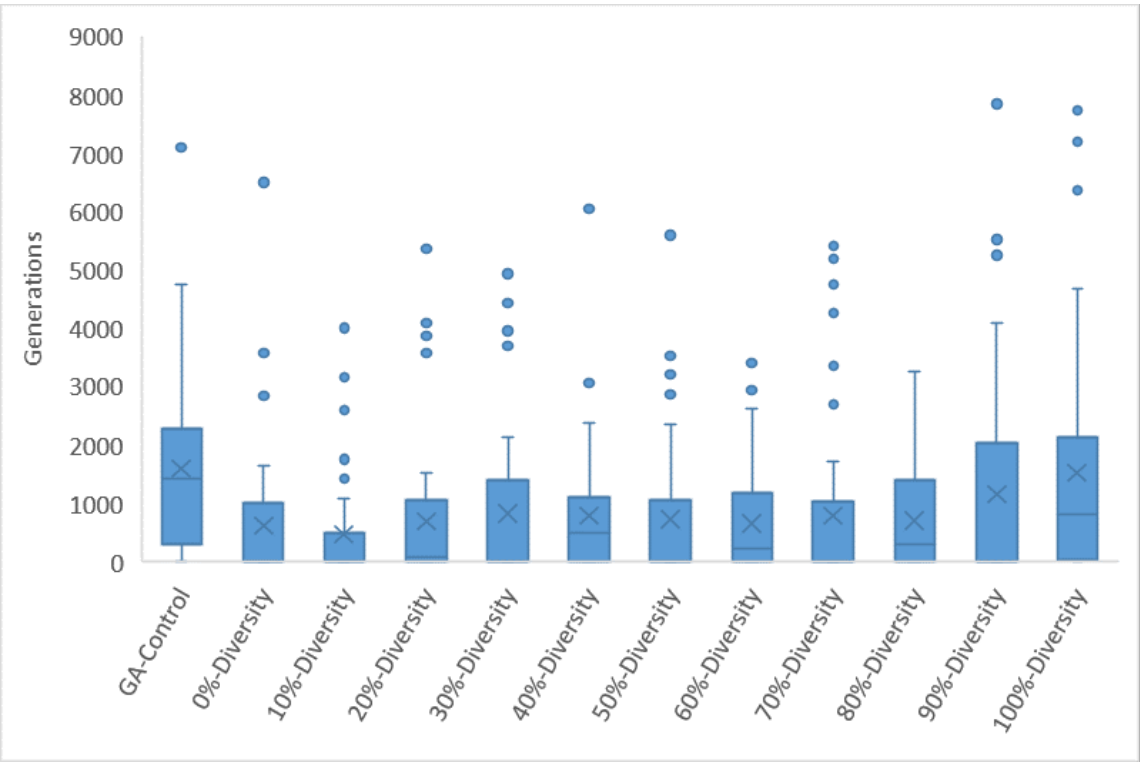

Figure 6.3: The number of generations of each version of the transferred population for solving the 7.2ca problem. The $GA(T)$ solver must add one element of knowledge to solve the T problem. The 10%-Diversity version of the transferred population shows a smaller number of generations compared to other versions. The 100%- Diversity version of the transferred population shows a high number of generations solving the T problem compared to other versions of the transferred population.

Table 6.2: The pairwise Mann-Whitney U test results of the first target problem (7.2ca). The $GA(T)$ solver must add one element of knowledge to find the optimal solution to the T problem. The results show how different each version is from the other version. This test yields p-value. For our study, if the p-value of the version of the transferred population < 0.05 indicates the average number of generations is a significant difference and it is bolded.

	0%-Diversity	10%-Diversity	100%-Diversity	20%-Diversity	30%-Diversity	40%-Diversity	50%-Diversity	60%-Diversity	70%-Diversity	80%-Diversity	90%-Diversity
10%-Diversity	0.95182	-	-	-	-	-	-	-	-	-	-
100%-Diversity	**0.00033**	**0.00010**	-	-	-	-	-	-	-	-	-
20%-Diversity	0.73563	0.73731	**0.00078**	-	-	-	-	-	-	-	-
30%-Diversity	0.72017	0.73561	**0.00145**	0.99716	-	-	-	-	-	-	-
40%-Diversity	0.28215	0.15845	**0.01571**	0.36607	0.47483	-	-	-	-	-	-
50%-Diversity	0.63292	0.61002	**0.00180**	0.91876	0.88466	0.53500	-	-	-	-	-
60%-Diversity	0.24345	0.25457	**0.00452**	0.45735	0.47221	0.83663	0.62959	-	-	-	-
70%-Diversity	0.68270	0.62109	**0.00173**	0.91624	0.87955	0.48236	0.96097	0.54868	-	-	-
80%-Generation	0.10878	0.07833	**0.02053**	0.18937	0.26041	0.79352	0.32847	0.60386	0.28038	-	-
90%-Diversity	0.24573	0.25173	**0.01892**	0.44135	0.35234	0.87570	0.50893	0.78389	0.52295	0.87327	-
Ga-Control	**1.706**	**6.707**	0.26104	**1.205**	**7.605**	**0.00023**	**3.705**	**6.905**	**3.605**	**0.00029**	**0.00396**

7.2ca problem. The $GA(T)$ solver must look for a solution by adding one element of knowledge and counting how many generations the $GA(T)$ solver took to find the optimal solution. Table 6.2 values show how different each version is from the other.

6.5.2 Second Target Problem

The second target problem 7.2cb is the problem in which the target solver must solve the T problem by adding four elements of knowledge to find the optimal solution of the T problem. As we mentioned before, we counted how many generations the $GA(T)$ solver took to find the optimal solution. The following graph Figure 6.4 is a description of how many generations the $GA(T)$ took to find the solution.

To make our study statistically sound, we ran a multiple comparisons test called the pairwise Mann Whitney U test of the 11 versions of the transferred population we created, as shown in Table 6.3.

Table 6.3: The pairwise Mann-Whitney U test results of the second target problem (7.2cb). The $GA(T)$ solver must add four elements of knowledge to find the optimal solution to the T problem. The results show how different each version is from the other version. This test yields p-value. For our study, if the p-value of the version of the transferred population < 0.05 indicates the average number of generations is a significant difference and it is bolded.

	0%-Diversity	10%-Diversity	100%-Diversity	20%-Diversity	30%-Diversity	40%-Diversity	50%-Diversity	60%-Diversity	70%-Diversity	80%-Diversity	90%-Diversity
10%-Diversity	0.1690	-	-	-	-	-	-	-	-	-	-
100%-Diversity	0.0515	**0.0023**	-	-	-	-	-	-	-	-	-
20%-Diversity	0.2761	0.6791	**0.0063**	-	-	-	-	-	-	-	-
30%-Diversity	0.9862	0.1712	0.1030	0.2776	-	-	-	-	-	-	-
40%-Diversity	0.7960	0.1487	0.1487	0.2185	0.8362	-	-	-	-	-	-
50%-Diversity	0.7097	0.1411	0.1045	0.2172	0.9313	0.8281	-	-	-	-	-
60%-Diversity	0.6172	0.0932	0.1477	0.1701	0.6172	0.8550	0.9862	-	-	-	-
70%-Diversity	0.7960	0.1467	0.1105	0.2454	0.9286	0.8496	1.0000	0.7097	-	-	-
80%-Diversity	0.8496	0.1420	0.1225	0.1891	0.8254	0.9670	0.9588	0.8767	0.9725	-	-
90%-Diversity	0.7097	0.0788	0.2250	0.1200	0.6343	0.7722	0.8496	0.9341	0.7226	0.7987	-
Ga-Control	0.0836	**0.0032**	0.9698	**0.0043**	0.0759	0.1285	0.1496	0.1997	0.1192	0.1168	0.1938

Table 6.3 shows the p-values of the 11 versions of the transferred population for solving the 7.2cb problem. The $GA(T)$ solver must look for a solution by adding four elements of knowledge and counting how many generations the $GA(T)$ solver took to find the optimal solution. Table 6.3 values show how different each version is from the other.

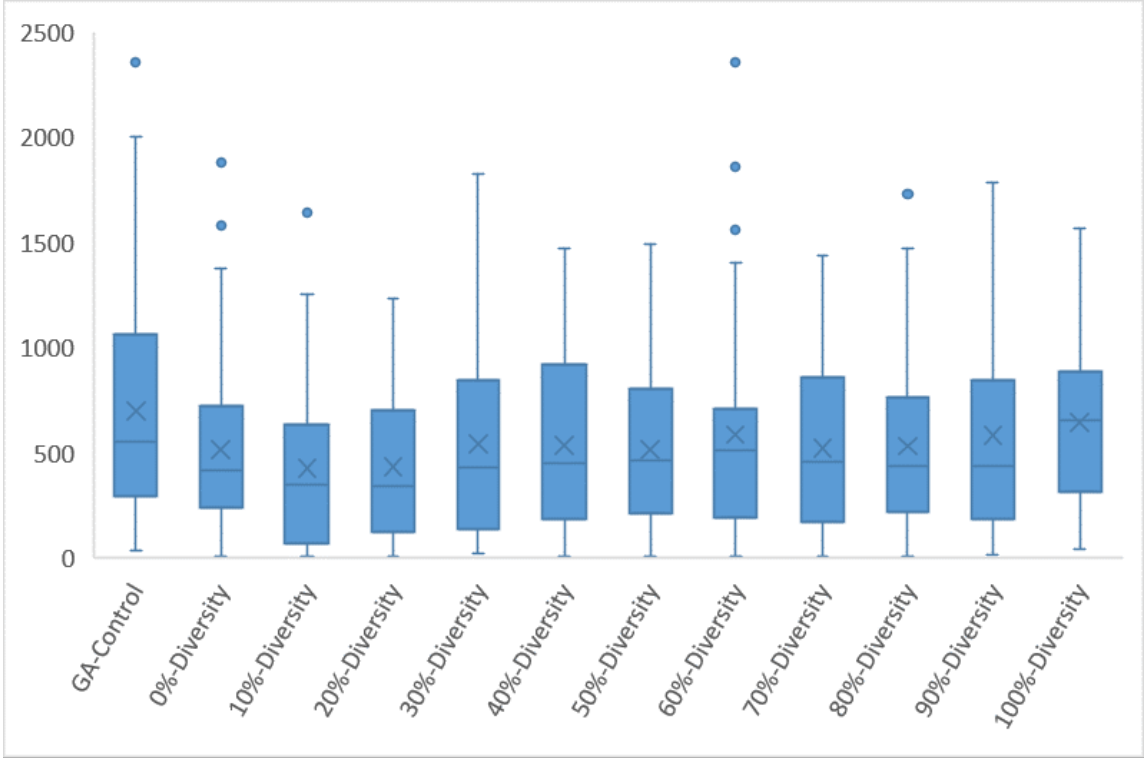

Figure 6.4: The number of generations of each version of the transferred population for solving the 7.2cb problem. The $GA(T)$ solver must have four elements of knowledge to solve the T problem. The 10%-Diversity and 20%-Diversity versions of the transferred population show a lower number of generation compared to other versions. The 100%-Diversity version of the transferred population shows a high number of generations solving the T problem compared to other versions of the transferred population.

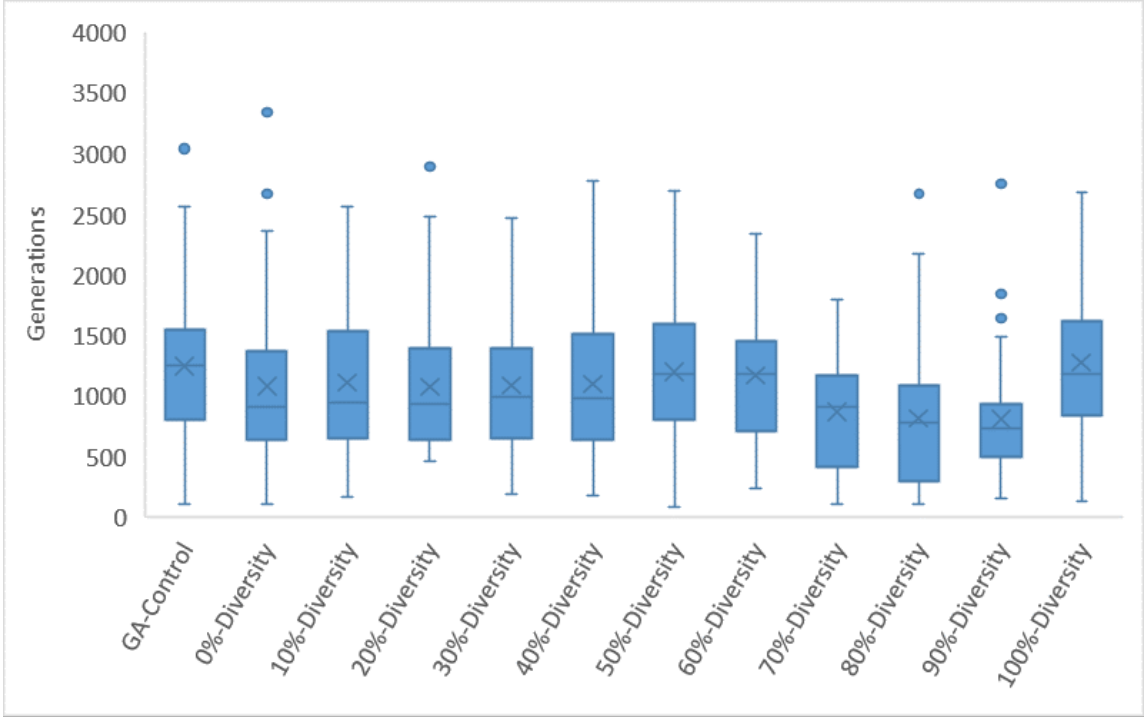

Figure 6.5: The number of generations of each version of the transferred population for solving the 7.2cc problem. The $GA(T)$ solver must add eight elements of knowledge to solve the T problem. The 70%-Diversity, 80%-Diversity, and 90%-Diversity versions of the transferred populations show a smaller number of generations compared to other versions. The 100%-Diversity version of the transferred population shows a high number of generations of solving the T problem compared to other versions of the transferred population.

6.5.3 Third Target Problem

The third target problem 7.2cc is the problem in which the target solver $GA(T)$ must find the optimal solution of the T problem by adding eight elements of knowledge. As we mentioned before, we counted how many generations the $GA(T)$ took to find the optimal solution. The following graph Figure 6.5 is a description of how many generations the GA(T) took to find the solution.

To make our study statistically sound, we ran a multiple comparisons test called the pairwise Mann-Whitney U test of the 11 versions of the transferred population we created, as shown in Table 6.4.

Table 6.4 shows the p-values of the 11 versions of the transferred population for solving the 7.2cc problem. The $GA(T)$ solver must look for a solution by adding eight elements of knowledge and counting how many generations the $GA(T)$ solver took to find the optimal solution. Table 6.4 values show how different each version is from the other.

Table 6.4: The pairwise Mann-Whitney U test results of the third target problem (7.2cc). The $GA(T)$ solver must add eight elements of knowledge to find the optimal solution to the T problem. The results show how different each version is from the other version. This test yields p-value. For our study, if the p-value of the version of the transferred population < 0.05 indicates the average number of generations is a significant difference and it is bolded.

	0%-Diversity	10%-Diversity	100%-Diversity	20%-Diversity	30%-Diversity	40%-Diversity	50%-Diversity	60%-Diversity	70%-Diversity	80%-Diversity	90%-Diversity
10%-Diversity	0.79069	-	-	-	-	-	-	-	-	-	-
100%-Diversity	0.06221	0.09056	-	-	-	-	-	-	-	-	-
20%-Diversity	0.78804	0.98350	0.05400	-	-	-	-	-	-	-	-
30%-Diversity	0.62451	0.87130	0.11441	0.75115	-	-	-	-	-	-	-
40%-Diversity	0.79335	0.92585	0.09802	0.95053	0.8307722	-	-	-	-	-	-
50%-Diversity	0.16903	0.28372	0.64664	0.21464	0.38317	0.27757	-	-	-	-	-
60%-Diversity	0.17996	0.29153	0.59315	0.17553	0.31417	0.27910	0.99175	-	-	-	-
70%-Diversity	0.19142	0.10897	**0.00119**	0.11760	**0.06367**	0.9942	**0.00763**	**0.00259**	-	-	-
80%-Diversity	**0.04027**	**0.02111**	**0.00014**	**0.01691**	**0.00863**	**0.02312**	**0.00132**	**0.00039**	0.41991	-	-
90%-Diversity	**0.02963**	**0.01086**	**2.205**	**0.00328**	**0.00293**	**0.01385**	**0.00036**	**0.00014**	0.38129	0.98075	-
Ga-Control	**0.04748**	0.12762	0.93955	0.05530	0.13738	0.09732	0.66907	0.53043	**0.00130**	**0.00015**	**1.905**

6.6 Discussion

The performance of the $GA(T)$ depends on the combinations of the initial transferred population. The transferred population must have some amount of old knowledge and population diversity. As we see from the results, the amount of the two components of the transferred population varies between different problems. For example, by looking at the first target problem, we determined that a small amount of population diversity was enough to find the optimal solution easily. Also, the same case happens for the second target problem. On the other hand, by looking at the third target problem , we determined that a huge amount 70% or 90% of population diversity was required to solve the problem.

The number of components of the transferred population required to solve a problem depends on the relationship R between the S and T problems. For example, in the introduction (Section 9.2) of this paper, we illustrated the idea of TL by using a tennis player. If the player wants to learn other activities similar to the tennis game, the player is more likely to adapt to the other game easily. But if the other game is completely different, like football or soccer, the player may face some difficulty.

We can think about the relationship R between the S and T problems as if both problems are related to each other and if they are drawn from the same domain. If this is the case, the amount of old knowledge must exceed the amount of the population diversity of the transferred population. But if the two problems are drawn from different domains, then the amount of the population diversity must exceed the amount of the old knowledge of the transferred population by 70% or more.

The right combinations of the transferred population must follow the classic exploitation and exploration strategy [21]. The exploitation can be performed by transferring the best individual of the S problems. The best individual represents the knowledge that the solver

knows about solving the S problem. The exploration can also be performed by promoting the amount of random population or diversity. The relationship R can be discovered by analyzing or assuming how close the S and T problems are to each other.

No matter which case of the relationship R between S and T problems are, we have either close or huge difference relationship. The transferred population must be a combination of previous knowledge and population diversity. The portion of each element of the transferred population is depending on how R is.

6.7 Conclusion

This paper studies how to find the right combinations of the transferred population content (knowledge and population diversity) and what factors may affect the performance of the $GA(T)$ solver. Analyzing the results from Section 9.5 proves that the relationships between the S and T are an important factor that has an effect on the performance of the $GA(T)$ solver. This paper also shows that the knowledge and population diversity of the transferred population must be available in the transferred population and varied regarding the differences in the relationship between the S and T.

We ran an experiment that has three different sets of the target problem. These problems vary between easy and hard cases. We measured the diversity and counted how many generations the $GA(T)$ solver took to find the optimal solution for each T problem.

The content of the transferred population makes a difference in speeding up the process of the $GA(T)$ of finding the optimal solution. The experiment and the results showed that the relationship R between the S and T is the key element that can quantify the amount of knowledge and diversity the transferred population must acquire.

Acknowledgment

This study is supported by Al Baha University, Al Baha, Saudi Arabia, and (SACM) Saudi Arabian Cultural Mission. I would like to thanks my supervisor Dr. Robert B. Heckendorn for his advice in this study.

Chapter 7

Analysis of the Effect of Population Content on Transfer Learning in Evolutionary Algorithms

This chapter discusses the fifth objective, which is "Analyze the effect on the content of the transferred population". This chapter studies the following concerns:

1 : Prove that the shared information saved in the population after solving the source problem helps solve the target problem.

2 : Investigate what are the important parts of the information from the previous populations needed to solve the target problem.

3 : Study how critical is the relationship between the source S and the target T problems in the TL environment.

7.1 abstract

Transfer Learning (TL) is a metaheuristic used in machine learning. Transfer Learning is using information used to solve one problem to improve the process of solving a related problem. In this work, we employ Evolutionary Algorithms (EA) since they lend themselves naturally and easily to TL by using components of the population that the EA used to solve a source problem to help solve a related target problem. A key idea in Transfer Learning is defining the relationship between the source and target problem. We explore the question of what information from the solved source problem is necessary to transfer for efficient solution of the new target problem. While our results are in the context of Evolutionary Algorithms the simple experimental design may suggest more general properties of TL.

7.2 Introduction

Transfer Learning (TL) is a metaheuristic that has been shown successful in improving optimization performance in many areas of Machine Learning (ML) where multiple similar problems are being solved [69]. TL has been implemented with Neural Networks [42], Reinforcement Learning [66], Genetic Programming [17], Genetic Algorithms [35] and others.

Transfer learning occurs when information used to solve a **source problem**, S, is used to improve the process of solving a related **target problem**, T. The improvement could be measured in a variety of ways. For example, it could be speed to solve, robustness of solution, or accuracy of answer. A key idea in transfer learning is the **problem relationship**, R, between S and T, the details of which may or may not be known to the solver. How S is related to T and how "close" that relationship is affects the effectiveness of TL in leveraging information learned in solving S.

A big question for TL research is, given the relationship between source and target problem, what information from the solved source problem needs to be transfered for efficient solution of the target problem? Understanding this will help algorithm designers to select what information to pass to the process for solving the target problem and may help accelerate the solving process.

We will model a general problem whose specification allows us to both vary the difficulty and structure of the problem and also rigorously define the relationship between S and T. Our solver will be a genetic algorithm, a subclass of evolutionary algorithms, which gives us a way to easily measure information transfer and solution performance. At the same time this will inform not just TL practitioners in general, but those specifically using evolutionary computation approaches in a TL context.

The practical side of Transfer Learning is that it may reduce the time it takes to learn the new task. TL is also valuable for problems where training data may be limited for a specific problem, but combining data from many related problems helps to provide enough data to provide practical solutions and aids in generalization [69].

To examine this question and others we will employ a genetic algorithms [19]. **Genetic Algorithms** (GA) are population based stochastic optimization algorithms. They lend themselves naturally and easily to TL by using components of the population that the GA used to solve problem S to help solve problem T. For our study subsets and variations of the population used to solve the source problem are transferred to the population used to solve the target

problem. The properties of the transferred population can be easily quantified.

By using problems with controllable similarity, we can test the relationship between the kind of kinship between the source and target problem and the information needed to solve a new problem.

Part of what makes this an intriguing approach is that a GA uses its population to sample the space of potential solutions. If S and T are related then parts of the solution to T may have been stumbled upon in searching for the solution for S. Sampling the population over time or only in the final solution population are strategies that may give a leg up in solving T.

Specifically the questions we wish to address are:

- **Q1:** Does sharing of information saved in the population after solving the source problem help solve the target problem more quickly?
- **Q2:** What are the important parts of the information from the previous population needed?
- **Q3:** The relationship between the source and target problem is critical to transfer learning. If the problems are not related then transfer learning won't work and, in fact, may degrade solution time. What is the relationship between the similarity of source and target problem and the time to solve the problem?

We will construct a set of tunable functions that allow us to control the difficulty of problems and the relationships between them. We will test a variety of parameterized strategies for transferring data from the solution for the source problem to the solver for the target problem.

To answer the questions above we use genetic algorithms to solve the source problem, S, and the population to represent what is learned in the process of solving S. We then try various Transfer Learning strategies to transfer knowledge from the solver implemented as a Genetic Algorithm (GA) [21] for S to the solver (also a GA) for T (see Section 8.4). These are represented as various schemes for transferring measured parts of the solution population from S.

The results show that in many cases sharing some or all of the final S population reduced the time to solve T as measured by the number of generations (Q1). However, the relationships between the S problem and the T problem is critical in determining what information to pass and how well TL works (Q2, Q3).

We show as the S and T become less and less related, it becomes less and less advantageous to use Transfer Learning. At some point, it actually hinders the solution of T. We examined what information helped best and measure the effect of diversity on the effectiveness of Transfer Learning. We use this information to suggest algorithmic improvements.

This paper is structured as follows: Section 8.3 is the background describing related research. Section 8.4 is how we implemented our model to answer our questions. Section 8.5 are the experiments that we implemented and the results of our findings. Section 9.7 summarizes the questions we address and our conclusions.

7.3 Background

Transfer Learning is not a new topic, TL has been studied and applied in many Machine Learning areas. Transfer Learning has been used successfully in Neural Networks [42] and Deep Neural Networks [42]. [66] implemented Transfer Learning in the Reinforcement Machine Learning. Transfer Learning in Genetic Algorithms was studied by Koçer et al. [1,35].

7.3.0.1 General Transfer Learning research

Lu et al. [42] publish a survey paper addressing and reviewing many approaches to Transfer Learning and Neural Networks (NN). They conclude that the performance and quality of the NN was increased because of Transfer Learning relaxes the idea that the NN weights must be in the format of **independent and identically distributed** (i.i.d) what does this mean?

Transfer Learning has been implemented in Reinforcement Learning (RL) [66]. The authors studied and published a survey paper that describes four types of strategies for Transfer Learning:

- The simplest strategy is use source problem knowledge in the target problem, assuming the knowledge has been learned.
- Reinforcement agent can learn multiple tasks and use all tasks to finish one task.
- Build a library for storing tasks and use them in the transfer.
- Allow source task to be modified by users and use the modified version in the target task.

They tested their strategies on the standard RL benchmark: mountain car. They found out transferring knowledge between the RL task is possible also, these strategies may be combined

together or with speed up methods to solve the final task quickly. Note that none of these strategies guarantee a best way of transferring knowledge from source to target problem. The authors discussed the problem of **negative transfer** which occurs when transferring knowledge makes the target learning process slower. This might occur if the algorithm is mislead by previous solutions or the information interferes in some way with the learning of the new solution. This situation may happen when the source problem is entirely different than the target problem.

The authors addressed the question of "what information is transferred between tasks"? They described the category of information being transferred. The first category is low-level knowledge transferring like RL policy, goal state, start states, start variable, a reward function or Q function of action-value (these terms are reinforcement learning terminologies). The second category is higher-level knowledge, such as a particular action must be used in a specific situation or state.

7.3.0.2 Transfer learning and GAs

Koçer et al. studied transfer learning in genetic algorithms [1,35]. They proposed an algorithm that simplifies the use of Transfer Learning based on the Genetic Algorithm. They claimed some individuals who had poor fitness values for the source problem, seemed perfect or almost the perfect for the target problem. So, they built a pool to collect these individuals (best, middle, and worst) of each iteration. When this pool was full of selected individuals, the pool was transferred to the target problem. They tested their algorithm in finding network topology and weights.

In [35] the authors wanted to see if TL would increase the performance of the GA. They formed two similar fitness functions for the source and the target problem. They claimed in the conclusion, the performance of the transfer genetic algorithm will improve significantly if the Euclidean distance between source fitness function and target fitness function is below 10^{-3}. They performed their studies using a general GA scenario. They formed a scenario where the population was 300 and each individual is 60 bits.

As previous works have shown an improvement by using Transfer Learning, our study attempts to provide a case study of how the relationship between source and target affects the success or failure of TL and what should be transferred to best solve the target problem. Even though we are restricting ourselves to GAs because of the quantifiabilty of the information

transfer, we hope that our results provide insight into the general workings of Transfer Learning.

7.4 Methods

To answer our questions above, we create a model of Transfer Learning that allows us to control the relationship between the source and target problems and the amount and kind of data transferred. The solver was cast in the form of a Genetic Algorithm to find the optimum fitness. The basic idea of searching a function landscape for an optimal value using information in a population of potential solutions is embodied by this model.

Our test problems are in a general class of landscapes defined over bit space (bit strings) that we will try to maximize. In the context of Genetic Algorithms, this is referred to as the **fitness function**. Similar ideas of constructing a function over bits and composed of subfunctions of bits can be found in the work of Melanine Mitchell's Royal Road Functions [45] and in Watson and Pollack's HIFF functions [72]. This was generalized as Embedded Functions in [29].

Consider an n bit function that is the sum of m equal non-overlapping subsections of s bits. Let $n = m * s$. Define the fitness $f : \mathcal{B}^n \rightarrow \mathbb{R}$ as:

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1])$$

where $a_i \in \mathbb{R}$ are weights, functions $g_i : \mathcal{B}^s \rightarrow \mathbb{R}$, $x \in \mathcal{B}^n$, and $x[a, b]$ extracts bits in positions a through b from bit string x.

By defining the fitness function structure this way, $\vec{a}$ becomes a tunable parameter representing the importance of each g_i from a fixed vector of non-overlapping g functions: $G = [g_0, \; g_1, \; g_2, \; \cdots, \; g_{m-1}]$. That is, f is a linear combination of subfunctions over disjoint subsets of bits [1] . The similarity of two functions $f_{\vec{a}_1}$ and $f_{\vec{a}_2}$ can be measured by the similarity of $\vec{a}_1$ and $\vec{a}_2$.

In our experiments we will initially restrict the values in $\vec{a}$ to $a_i \in \{0, 1\}$ (We will discuss some of the consequences from this choice at the end of this section). $\vec{a}$ can be thought of as choosing which g_i matter in the evaluation of f. In a biological analogy, this might be what snipits of knowledge are necessary to survive and what are not. The relationship R between $f_{\vec{a}_1}$ and $f_{\vec{a}_2}$ can be partially measured in this case by using variants of the Hamming distance between $\vec{a}_1$ and $\vec{a}_2$. Other distance measures such as Euclidean distance for a broader class of $\vec{a}$ could also be used.

[1] Our work extends to the more difficult class of overlapping subsets of bits and nonlinear epistatic interactions, but that is not discussed in this paper.

In our experiments we will also fix the g_i functions as classic **Deceptive Functions** [26]. These functions are characterized by being defined over a s bit space in such a way that the function maximum is at a bit string b_{max}, but, in a misleading fashion, the function value grows as the Hamming distance between the domain value the complement of b_{max} becomes smaller. For example, assuming $\mathrm{bc}(b)$ is the number of 1 bits in the binary string b, here is a **deceptive function** with b_{max} equal to all 1's:

$$g(b) = \begin{cases} n & \mathrm{bc}(b) = n \\ n - 1 - \mathrm{bc}(b) & \text{otherwise} \end{cases}$$

These functions, by their deceptive nature, are notoriously difficult to solve.

Our experiments can now be formulated as solving a source fitness function $f_{\vec{a}_s}$ and then resolving on a target fitness function $f_{\vec{a}_t}$. The relation between the functions is a function of the $\vec{a}_s$ and $\vec{a}_t$.

Table 7.1: The four quadrants of the relationship between source and target problems. S and T are the bitstring representions of $\vec{a}$ for the source and target problems. $\overline{S}$, is the one's complement of the the bitstring S which represents the set of subfunctions that have a zero weight. Similarly for $\overline{T}$. The table shows how inclusion and exclusion of subfunctions in the fitness between source and target problems adds and subtracts knowledge requirements for successfully maximizing the fitness.

	S	$\overline{S}$
T	Conserved Knowledge $S \wedge T$	New Knowledge $\overline{S} \wedge T$
$\overline{T}$	Obsolete Knowledge $S \wedge \overline{T}$	Irrelevant Knowledge $\overline{S} \wedge \overline{T}$

A motivation for using a $\{0, 1\}$ vector is that it models the acquisition and loss of knowledge, since in order to get maximum fitness the function must solve each g_i where $a_i > 0$. Knowing the bit values that will maximize a specific g_i whose $a_i > 0$ is "knowledge" of how to get a better fitness in general. In this simple model we won't model rewarding for forgetting knowledge or replacing knowledge with new knowledge. But it could be easily done.

For these experiments the vector $\vec{a}$ can be represented as a bitstring. If we let S be the bitstring representation of $\vec{a_s}$ and T be the bitstring representation of $\vec{a_t}$ then ideas such as "new knowledge needed" can be represented by the bitwise logical expression $\overline{S} \wedge T$, (See Table 7.1). That is, the 1 bits in $\overline{S} \wedge T$ indicate the added subfunctions that must be solved in moving from source problem to target problem. $S \wedge \overline{T}$ indicates which functions were needed in the

source problem, but are no longer needed. Similarly for the other quadrants in Table 7.1. The number of subfunctions involved is simply the number of 1 bits which can be counted by using the bitcount also known as the unitation function, bc.

For example, let:

$$\vec{a_s} = (1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0) \tag{7.1a}$$

$$\vec{a_t} = (0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1) \tag{7.1b}$$

then:

Conserved Knowledge: $\mathrm{bc}(S \wedge T) = 1$ subfunctions.

New Knowledge: $\mathrm{bc}(\overline{S} \wedge T) = 4$ subfunctions.

Obsolete Knowledge: $\mathrm{bc}(S \wedge \overline{T}) = 3$ subfunctions.

Irrelevant Knowledge: $\mathrm{bc}(\overline{S} \wedge \overline{T}) = 2$ subfunctions.

We will use a Genetic Algorithm [19, 26] to solve the problems. A **Genetic Algorithm** (GA) is a type of Evolutionary Algorithm, which is an algorithm inspired by biological evolution. A GA has several key components. A population of potential solutions, in our case bit strings from $\mathcal{B}^n$. Imperfect copies of the individuals from the population will be made using mutation and mixing operators. Selective pressure will be applied to force an enrichment in the population with strings with higher fitness. This makes an optimization algorithm for the fitness function this is similar to that envisioned by Charles Darwin [12]. In a **Generational Genetic Algorithm** the population is "overwritten" with a new population. Each replacement of the population referred to as a **generation**. So solution time is measured in generations until the optimum is found. Table 7.2 describes the Genetic Algorithm parameters used in this paper.

All of our experiments will be run on functions defined as described above using a vector $\vec{a}$. An individual in the GA population is a string of 40 bits. The vector represents the coefficients for the 10 subfunctions, g_i, defined over 4 bits each. The relationship between the source and target problem will be controlled by the similarity of the vectors $\vec{a_s}$ and $\vec{a_t}$. The population consists of 1000 individuals.

All of the following experiments apply this theme: To test Transfer Learning, two fitness functions will be chosen. When the GA finds the best solution to the source problem, the number of generations will be recorded. Some part or all of the final population will be passed to the GA solving the target problem and the number of generations will again be recorded.

For each problem pair, 50 replicates of each experiment are performed for statistical purposes.

Table 7.2: Genetic Algorithm Parameters

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	1000
Mutation rate (per bit)	0.1
Crossover rate	0.05
Type of crossover	Uniform crossover
Tournament Size	5

7.5 Experiments

In order to answer our questions above, we have developed three sets of experiments, and these experiments have a number of parts. We begin by evaluating the effectiveness of implementing Transfer Learning in a GA to improve performance in a set of related problems. In the second experiments, we examine what information needs to be passed from source solver to target solver to best improve performance. In the final experiments we look at how the relationship between source and target problems influences the effectiveness of Transfer Learning.

7.5.1 Experiment 1: Transfer Learning in a GA

This experiment has two sections, the aim of Section A is to prove that TL can improve the performance of GA. After we proved that, TL improve the performance of the GA. The question is, how good is this over all function? To answer this question, we ran Section B.

Section A:

For this experiment we choose function vectors for the source and target problems, $\vec{a_s}$ and $\vec{a_t}$, that are all 1's except for a 0 in a single location. 50 replicates of our experiments are performed for statistical purposes. For example:

$$\vec{a_s} = (0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2}$$

$$\vec{a_t} = (1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.3}$$

We begin by setting a base experiment to answer the question does TL work in the setting of our experiment. Specifically we are asking:

Q1: Does sharing of information saved in the population after solving the source problem help solve the target problem? In particular, does it solve the problem more quickly? If it does not solve the problem, are the answers computed in the same amount of time more fit?

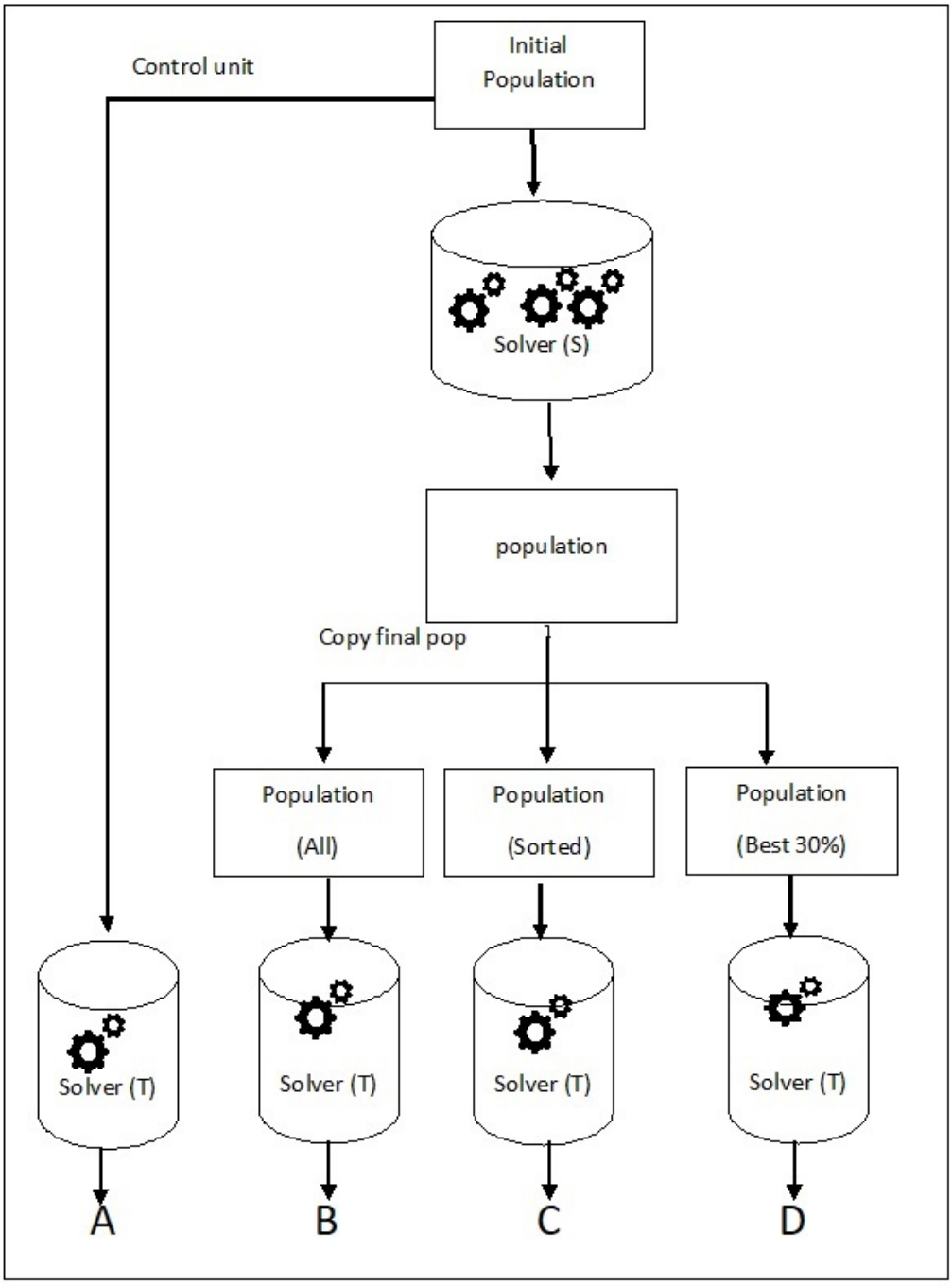

Figure 7.1: A random initial population is passed to an GA to solve the source problem. The final population from that optimization will then be copied into three modified copies. These are: All of the final population from the Source problem solution, Sorted final population (this should make no difference), and Best 30% of the final population with the remaining 70% being random. These copies will be the initial populations in the GA solving the target problem. A control population consisting of the initial population for the source problem GA is also tested to provide a control experiment for the performance of Transfer Learning.

In order to answer question one, we create four strategies labeled A, B, C, and D in Figure 7.1.

Strategy A, labeled as "NoTL", short for "No Transfer Learning", starts by solving problem T from scratch, that is from a random population. Since this does not use transfer learning, this is the control for our experiments.

Strategy B, labeled "All", we use the source problem final solution as the initial population of the target problem. The box plot labeled as "All" in Figure 7.2.

Strategy C, labeled "Sorted", We sort the population based on their fitness values and pass it to the target problem. Also, our measurement value is recorded. We expect this strategy will not make much difference in the outcome from the unsorted strategy.

Strategy D, labeled "30%", transfers the best 30% of the source problem S final population. The remaining 70% of the population is generated randomly.

Figure 7.2 shows the distribution of the number of generations it took to find the optimal solution as a quartile box plot.

As we expected Table 7.3 and Figure 7.2 do not show a significant difference between "All" and "sorted" strategies. The p-value between "All" and "sorted" strategies as shows in Table 7.3 is equal to 0.61. This high value is enough to suggest that our expectation was right.

Figure 7.2 is a box plot graph that represents the results of the experiment that we ran fifty times. The three above strategies labeled All, sorted, and best 30% are represented Transfer learning experiments and the box plot labeled as "NoTL" is represented as the source strategy. The ranges of the Transfer Learning strategies are less than the ranges of "NoTL" strategy including Transfer Learning experiments outliers. The minimum of the Transfer Learning is the same but is different in the "NoTL" strategy. The Transfer Learning strategies show a close mean whereas it is totally different from the "NoTL" strategy. In our experiments, the Transfer Learning experiment finds the best solution within 25% of the number of generations of the "NoTL" strategy.

Figure 7.2 shows the distribution of the number of generations it took to find the solution when the population is initialized to a random chromosome as a quartile box plot.

We used the Kruskal-Wallis and non-parametric statistic tests. With a p-value equal to 2.2×10^{-16} we have enough statistical evidence that the Transfer Learning had a sufficient improvement in the Genetic Algorithms. Table 7.3 is the pairwise Mann-Whitney U-tests, which is a multiple comparison test of the first four experiments we did:

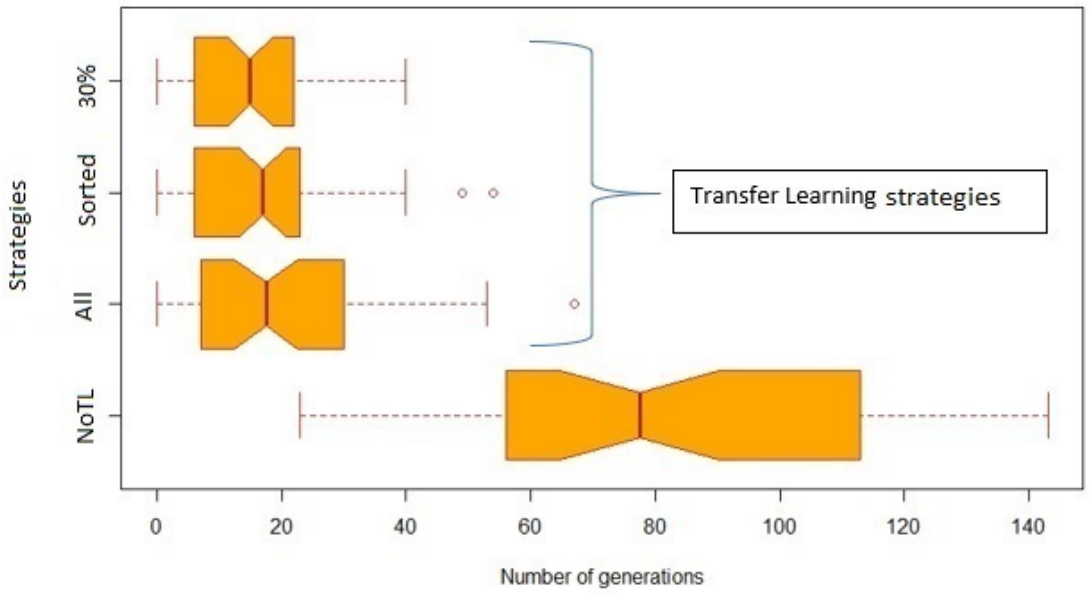

Figure 7.2: Transfer Learning Experiment: This quartile box plot shows the results of the transfer learning experiments compared to source experiment which represent non-transfer learning experiment. The results of each transfer learning experiment is less than the results of non-transfer learning experiment.

Table 7.3: This is the significance level as computed by the Pairwise Mann-Whitney U-tests. A low value indicates the likelihood that the two distributions are different.

	NoTL	All	Sorted
All	1.215	-	-
Sorted	< 216	0.61	-
Best 30%	< 216	0.38	0.73

Table 7.3 shows how source experiment is different than the Transfer Learning experiments. The difference is huge between source and (Target, sorted, and best 30%) which represent Transfer Learning experiments. Also, Table 7.3 shows the differences between Transfer Learning experiments are too close to call.

Table 7.4: Mean and standard deviation of the number of generations used by the GA to solve the target problem given the four different transferred populations in our experiments. Data averaged over 50 trials. Standard deviation in parentheses.

NoTL	All	Sorted	Best 30%
79.34(33.02)	19.42(15.97)	17.20(13.30)	15.80(11.89)

Table 7.4 shows the means and standard divisions values of each experiment. The differences between source experiments where the algorithms started from scratch and the Transfer Learn-

ing experiments where the knowledge had been transferred. These values indicate significant improvements between source and (All, sorted and 30% best) Transfer Learning experiments. Also, this table shows there is no big difference between "All" and "Sorted" experiments.

For the given source and target problem the Transfer Learning experiment improves the Genetic Algorithm process of finding the best solution by more than 60% compared with the source or with the Genetic Algorithm alone.

Section B: We aimed to answer the question of how good TL is for all functions? To answer this question, we ran experiment that tested the complete combinatorial set of S and T functions with {0 - 1} coefficients and the four relationships between S and T as shown in Table 7.1. We developed 286 functions pairs of source and target problems, also we classified these functions based on these four attributes of the relationships and counting up the number of occurrences of sub-functions in each category(as we will discuss next).

We performed this experiment by using the "Best 30%" strategy of transfer learning and "NoTL" as a control unit. Each function ran 50 iterations, and the number of iterations that the target solver took to find the solution was recorded. Then we ran correlations between the number of generations of TL strategies and the number of the (i,n,o,and c) knowledge. As describes in Table 7.5.

The length of each problem is 10 subfunctions. If we assume the order of these subfunctions is independent, we can pull them all. We have ordered them in the format of the first irrelevant knowledge (i), second new knowledge (n), third obsolete knowledge (o), and fourth conserved knowledge (c). They all sum in the abbreviation (inoc). For example,

$$\vec{a_s} = (i,\ i,\ n,\ n,\ o,\ o,\ o,\ c,\ c,\ c) \tag{7.2a}$$

$$\vec{a_t} = (i,\ i,\ n,\ n,\ o,\ o,\ o,\ c,\ c,\ c) \tag{7.2b}$$

$$\vec{a_s} = (0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2c}$$

$$\vec{a_t} = (0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1) \tag{7.2d}$$

The relationships are:

- 2 irrelevant knowledge.
- 2 new knowledge.
- 3 obsolete knowledge.

- 3 conserved knowledge.

As we mentioned above, we have 10 sub functions. we organized them as the following sections:

- 00 represents (irrelevant knowledge - i) first section.
- 01 represents (new knowledge - n) second section.
- 10 represents (obsolete knowledge - o) third section.
- 11 represents (conserved knowledge - c) fourth section.

The total number of 286 functions come from the following:
We must have combinations of the sections (i,n,o,and c) that sum up to 10 sub functions, and none of them is a negative number, because they can be zero.

for example:

3 + 2 + 3 + 2 = 10

i + n + o + c = 10

S : 000 - 00 - 111 - 11

T : 000 - 11 - 000 - 11

As we added three dividers to create four sets that represent (i,n,o, and c) knowledge, we now have a total of 13 things that take 3 at a time. So the 286 = combinations of (13,3).

This type of setting allows us to classify each function by its class. For example, how much new knowledge does a function has, or how much irrelevant knowledge other function has, and etc. Also, we can ask questions for example, what it takes to solve a problem that has a higher number of new knowledge. We can also find the correlations between the number of generations the target solver took to find the solution and the number (i,n,o, and c) knowledge.

Table 7.5: The correlation values of Section B experiment. The 286 functions source and target pairs run 50 times. Each time the number of generations of the TL strategy is recorded. On the left-hand side of this table is the TL strategy was used. We correlate the left side of this table with the number of influences of the (i,n,o, and c) knowledge relationships. The positive value indicates a positive correlation and the negative value indicates a negative correlation.

	Irrelevant Knowledge	New Knowledge	Obsolete Knowledge	Conserved Knowledge
NoTL	-0.032	0.028	-0.023	0.028
Best 30%	-0.005	0.259	-0.011	-0.241

Table 7.5 describes the correlation between the four relationships (i,n,o, and c) from Table 7.1, and number of generation the target solver took to find the solution of each strategy of TL ("NoTL" and the "Best 30%"). The correlations can be summarized as the following:

1. There is a negative correlation between the new knowledge relationship and the Best 30% strategy.

2. There is a positive correlation between the obsolete knowledge relationship and the Best 30% strategy.

3. There is a negative correlation between the irrelevant knowledge relationship and the Best 30% strategy.

4. There is a negative correlation between the conserved knowledge relationship and the Best 30% strategy.

The following scatter plots Figures (7.3 - 7.6) describe the type of correlation between each type of relationship and the Best 30% TL strategy.

7.5.2 Experiment 2: Damaged Population

The second experiment 7.5.2 was designed to answer question two which was what information should be transferred to the target problem. We created the following four problems. We asked the system to chose randomly a source problem and a target problem. We made sure the source problem is not like the target problem, and each attempt ran 50 times. This experiment has two sections (A & B). The source S and the target T problems of the A section are the same for the B section.

$$\vec{a_1} = (0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2ea}$$

$$\vec{a_2} = (1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2eb}$$

$$\vec{a_3} = (1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2ec}$$

$$\vec{a_4} = (1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2ed}$$

Part A, after the source problem S had finished its task of finding the best solution based on its fitness function the final population of the source problem was running through a damaged function. The damaged population is then used in the target problem to find the best solution

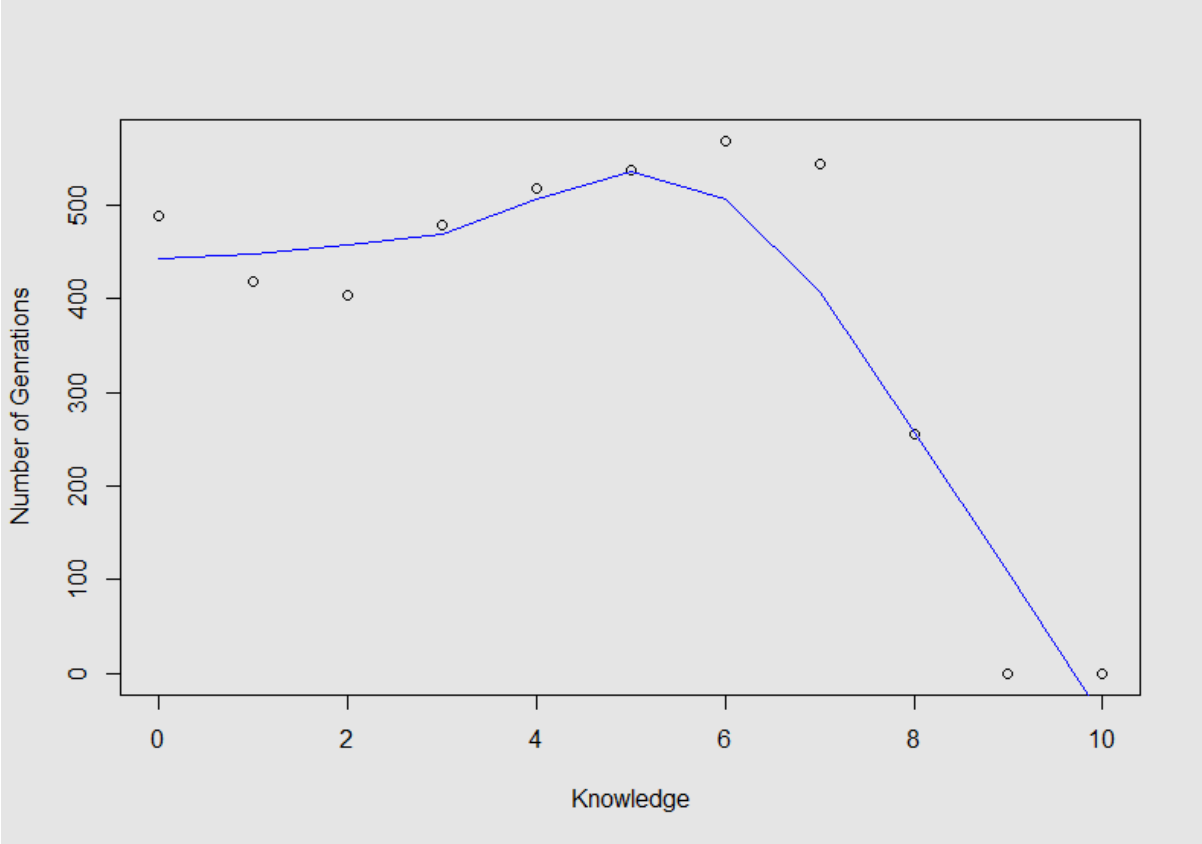

Figure 7.3: A scatter plot of irrelevant relationship type. This graph describes the distribution of the irrelevant knowledge relationship, and the Best 30% strategy of transfer learning. Each function of the 286 functions runs 50 times. The x-axis describes the knowledge, and the y-axis describes the average number of generations taken by the target solver to find the solution. As plotted in the graph the fit line decreases as we move to a high number of knowledges which indicates a negative correlation.

and count how many generations it takes to find the best individual. Figure 7.7 describes the situation of the source problem finishing its task and finding the best population. Those population will go to the target problem T through the damage function. The target problem will process its task of finding the best solution based on the damaged population of the source problem S. We count how many generations it takes to find the best solution.

Damage Function Damage function is the method we used to flip the individual's bits value. Each bit value flipped randomly to zero or one. Each time we damaged 10% of the chromosome size. We started from zero percent where there was no change and increased the damage process by 10% until we had 100% change of the chromosome.

Part B, the best individual of the source problem S copied as the size of the population which is 1000 individuals. The damage function takes a parameter p which is the probability that a given bit will be flipped. If $p = 0\%$, then that zero percent of the bits are not flipped and

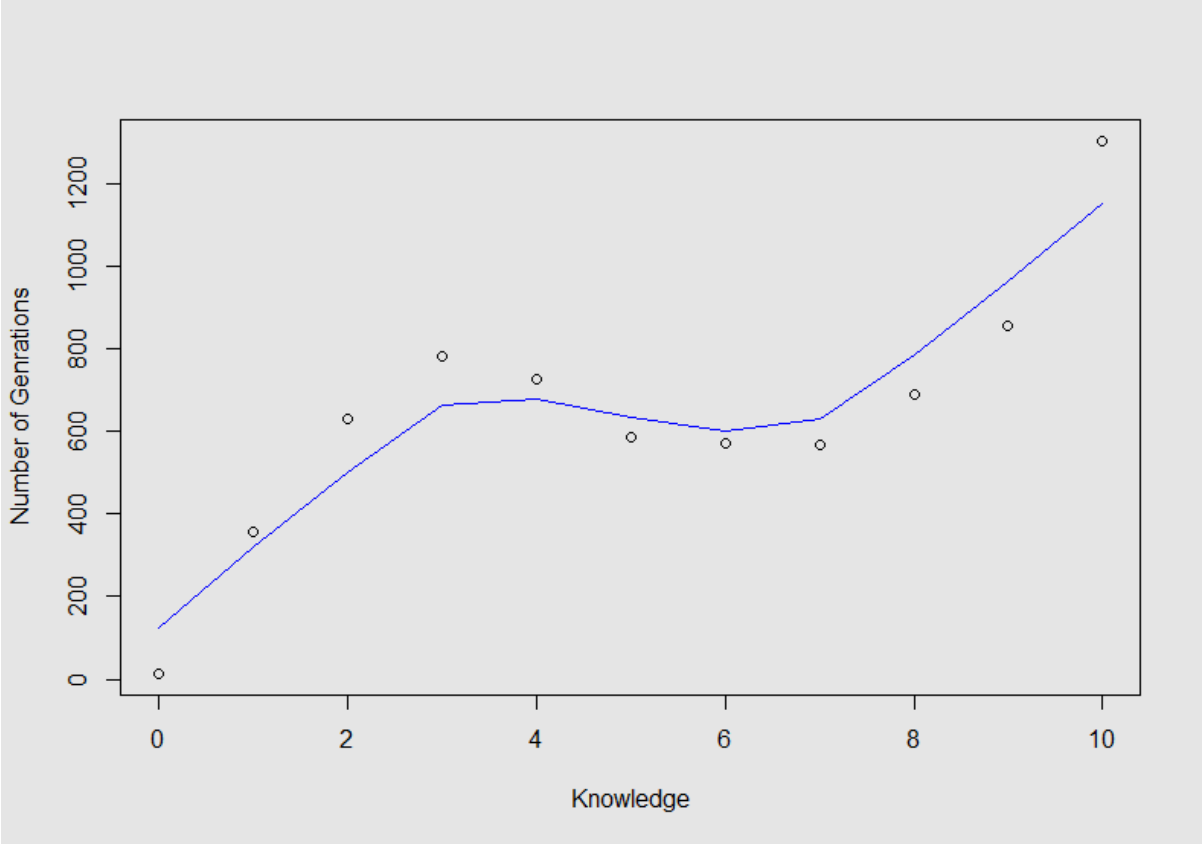

Figure 7.4: A scatter plot of new relationship type. This graph describes the distribution of the new knowledge relationship, and the Best 30% strategy of transfer learning. Each function of the 286 functions runs 50 times. The x-axis describes the knowledge, and the y-axis describes the average number of generations taken by the target solver to find the solution. As plotted in the graph the fit line increases as we move to a high number of knowledge which indicates a positive correlation.

the population remains untouched. If $p = 100\%$, the population returned is the 1's complement of the individuals in the population. If $p = 50\%$, then on average 50% of the bits are flipped.

In Part B of the experiment, we replicated the best individual in the population to fill the population. We then applied the damage function and used that population to attempt to solve problem T. Figure 7.8 shows the distribution of the number of generations needed to solve problem T.

Figure 7.8 can conclude this experiment as the best way to transfer knowledge to the target problem is the best individual. The graph shows a situation called the **Phase Transition** situation where there is a significant difference between zero percent damaged population and the rest of the percent's damaged population of the best individual.

The mean generations of finding the optimal solution are increasing periodically by going from low percent damaged to high percent damaged. Also, by comparing Figures 7.7, 7.8, and

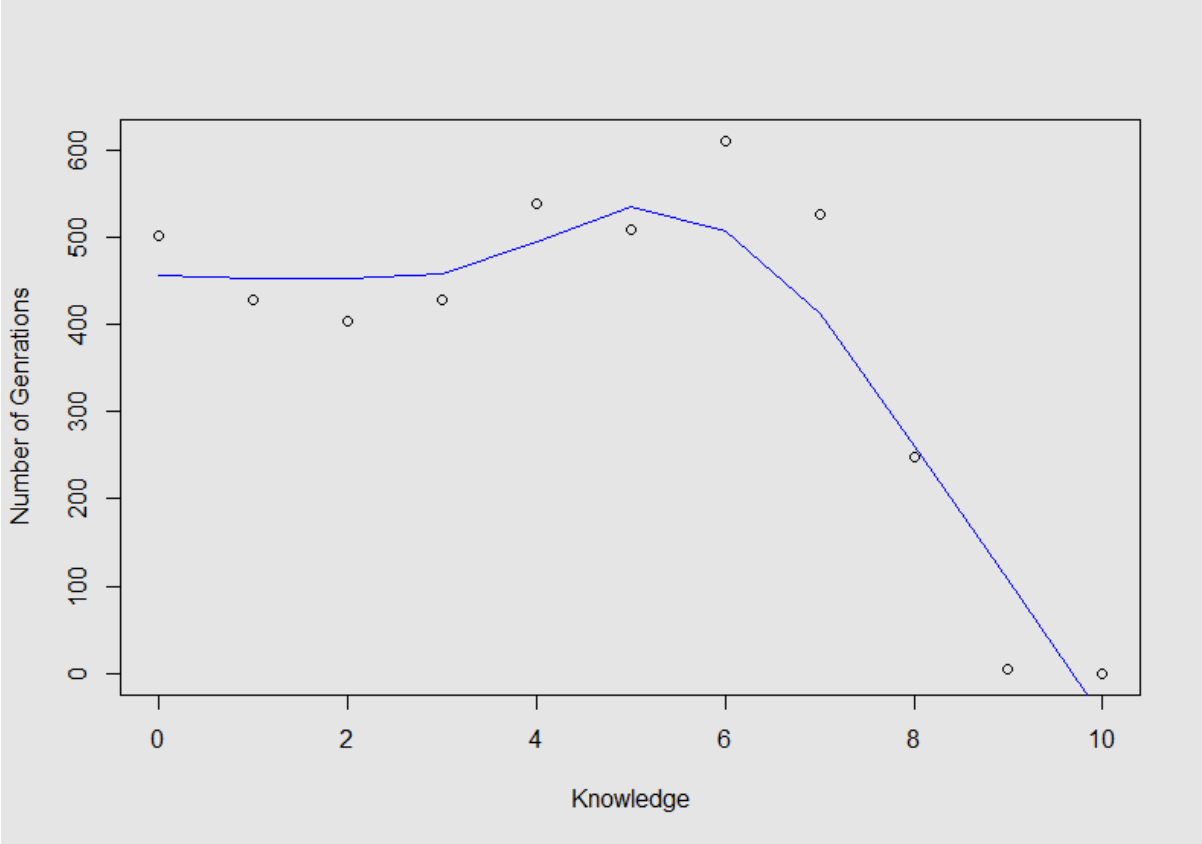

Figure 7.5: A scatter plot of obsolete relationship type. This graph describes the distribution of the obsolete knowledge relationship, and the Best 30% strategy of transfer learning. Each function of the 286 functions runs 50 times. The x-axis describes the knowledge, and the y-axis describes the average number of generations taken by the target solver to find the solution. As plotted in the graph the fit line decreases as we move to a high number of knowledge which indicates a negative correlation.

7.2 box plot labeled as "NoTL" the ranges look like each other. This gives us an indication of the best way of finding where is the best knowledge which is 0 percent damaged of the best individual population, as shown in Figure 7.8.

7.5.3 Experiment 3: The Relation between Source and Target

The third experiment 7.5.3 was designed to answer question three which was about the relationship between the source problem S and the target problem T. Discovering the relationships is a critical issue in Transfer Learning. It can indicate how well Transfer Learning will aid in solving the T problem can be done. For example, if there is little relationship between the source S problem and the target T problem, the process of the TL may dramatically degrade the benefits of TL. In order to answer our questions about relationships, Table 7.1 lays out the relationships between the source problem S and the target problem T. We design two experiments followed

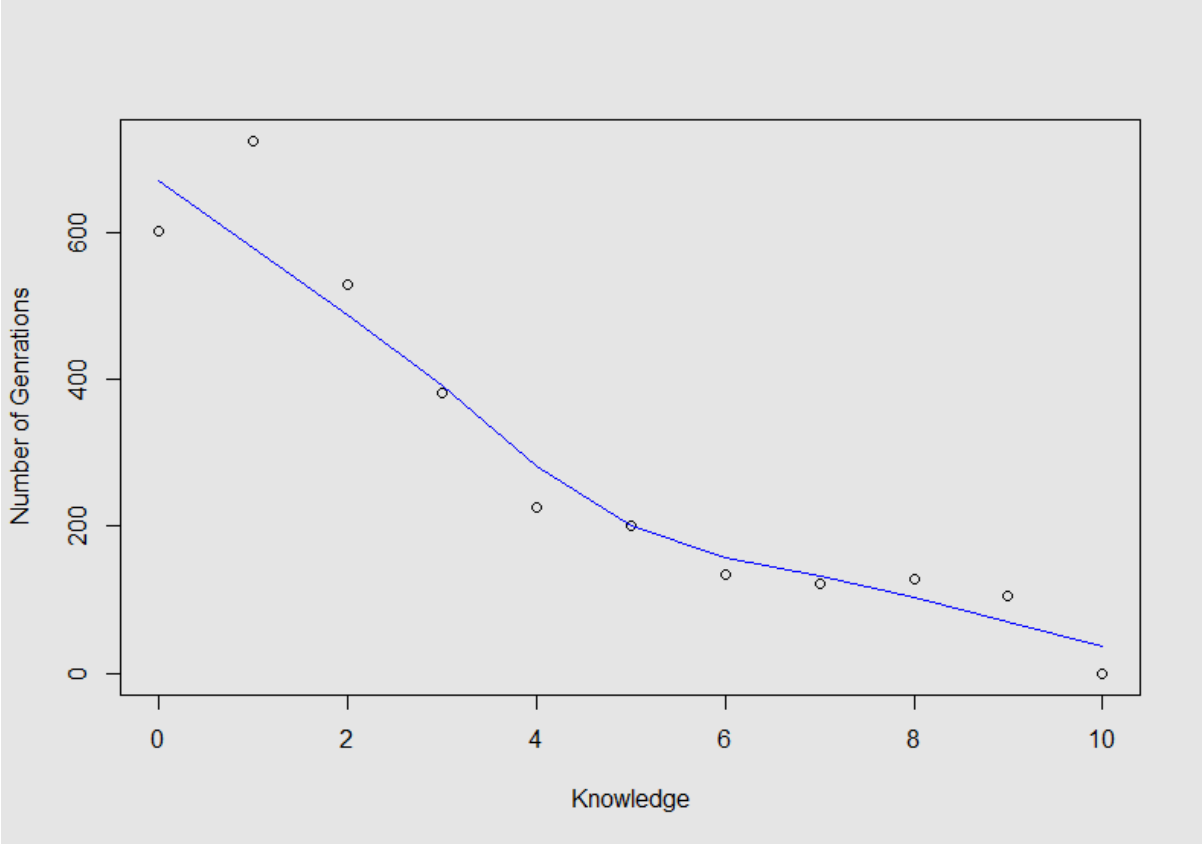

Figure 7.6: A scatter plot of conserved relationship type. This graph describes the distribution of the conserved knowledge relationship, and the Best 30% strategy of transfer learning. Each function of the 286 functions runs 50 times. The x-axis describes the knowledge, and the y-axis describes the average number of generations taken by the target solver to find the solution. As plotted in the graph the fit line decreases as we move to a high number of knowledge which indicates a negative correlation.

by figures that show the results.

We run the experiments using a GA with a bit-flipping mutation operator and two different crossover operations: two point crossover and uniform crossover.

The first try, we run the following experiment fifty times using four fitness functions and using two point crossover as the GA operation. The first experiment has two fitness functions and the second one has the other two fitness functions. We record how much generations each one takes to find the optimal solution. We plot the results in a box plots (Figure 7.9 and 7.10) to show the differences.

Problem 1, has the following fitness functions and the relationships:

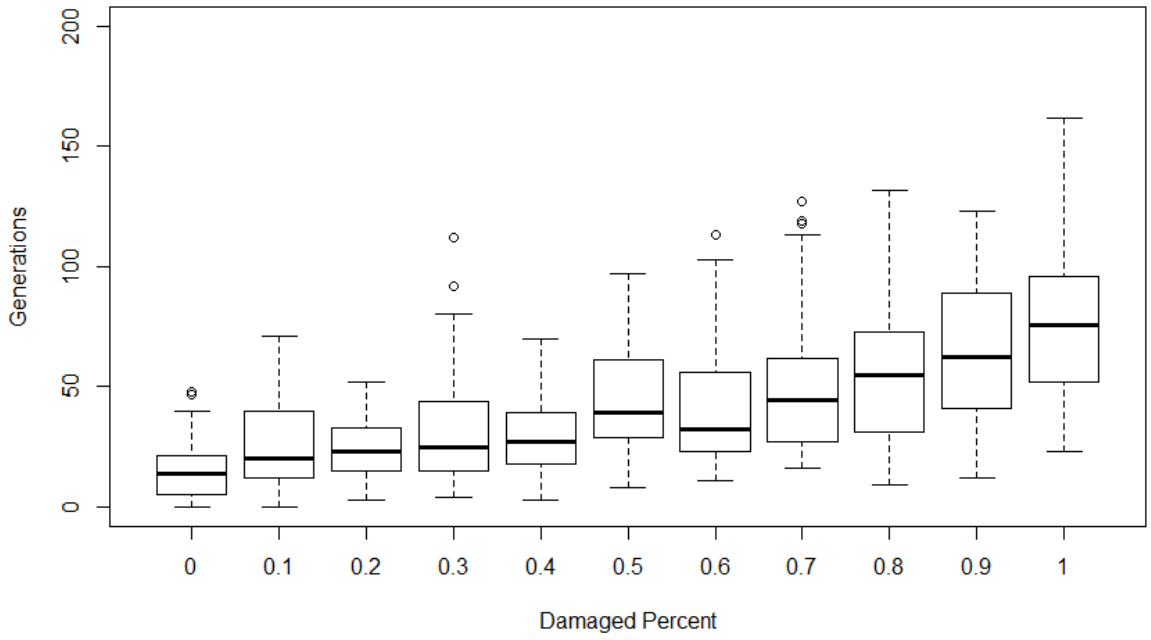

Figure 7.7: Result of all damaged population after the target problem has find the best solution. Each attempt runs 50 times.

$$\vec{a_s} = (1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1) \tag{7.2fa}$$

$$\vec{a_t} = (1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0) \tag{7.2fb}$$

- $\mathrm{bc}(\overline{S} \wedge T) = 2$ new knowledge.
- $\mathrm{bc}(S \wedge \overline{T}) = 2$ Obsolete knowledge.
- $\mathrm{bc}(S \wedge T) = 3$ Conserved knowledge.

Problem 2, has the following fitness functions and the relationships:

$$\vec{a_s} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1) \tag{7.2ga}$$

$$\vec{a_t} = (0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0) \tag{7.2gb}$$

- $\mathrm{bc}(\overline{S} \wedge T) = 4$ new knowledge.
- $\mathrm{bc}(S \wedge \overline{T}) = 4$ Obsolete knowledge.

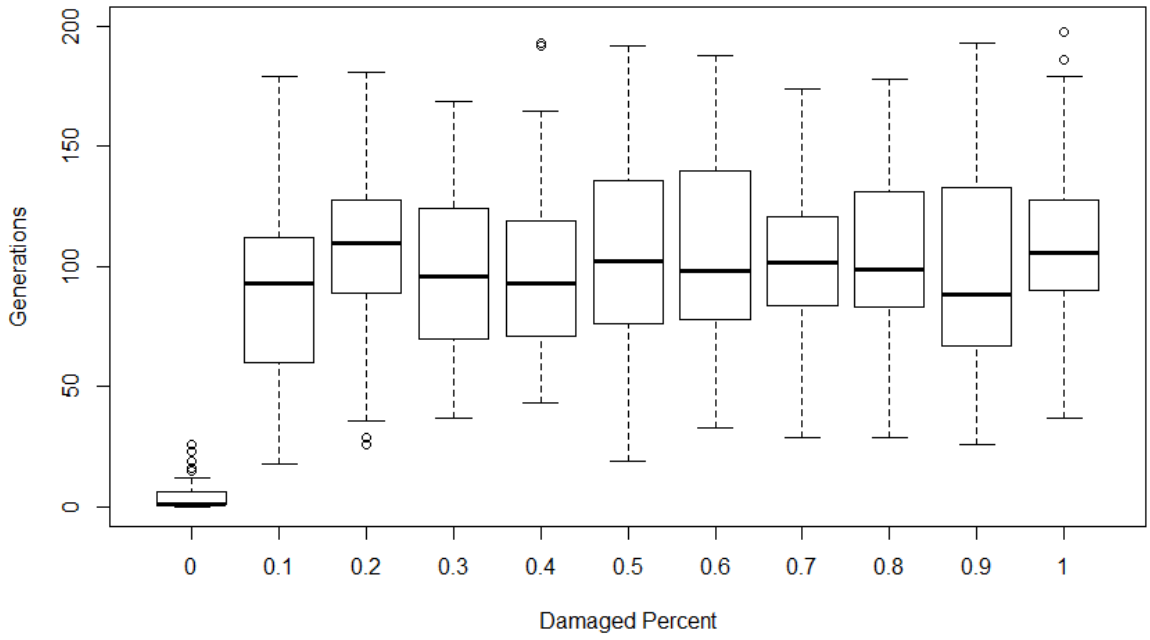

Figure 7.8: Result of all damaged population after the target problem has find the best solution. Each attempt runs 50 times. The 0 percent damaged population shows less number of generations compared to other damaged populations.

- $\text{bc}(S \wedge T) = 1$ Conserved knowledge.

According to [46] the situation in the Figures 7.9 and 7.10 is called **Hitchhiking**. Hitchhiking is a situation where a gene changes its value repeatedly. This change is forced by other nearby genes. When one is gene forced to change, any other nearby gene will tend to change, too. The forced change may happen when the mutation rate changes.

Figure 7.9 shows the plot of the two previous problems (1 & 2) above. The plot shows if the T problem learn new relationship $\text{bc}(\overline{S} \wedge T) =$ is close to the S problem by $(1, 2)$ new knowledge the experiment will not evolve many of generations to find the optimal solution as shown in the Figure 7.9 "Exp1:learn-New=2". Also if the T problem learn new relationship $\text{bc}(\overline{S} \wedge T) =$ is like 4 and above new knowledge or (subfunctions) to the S problem the experiment will evolve too many generations to find the optimal solution. Figure 7.9 "Exp2:Learn-New=4" shows the results.

Figure 7.10, shows the results of using a uniform cross over instead of two points cross over. We run the same experiments using uniform crossover operation of the GA.

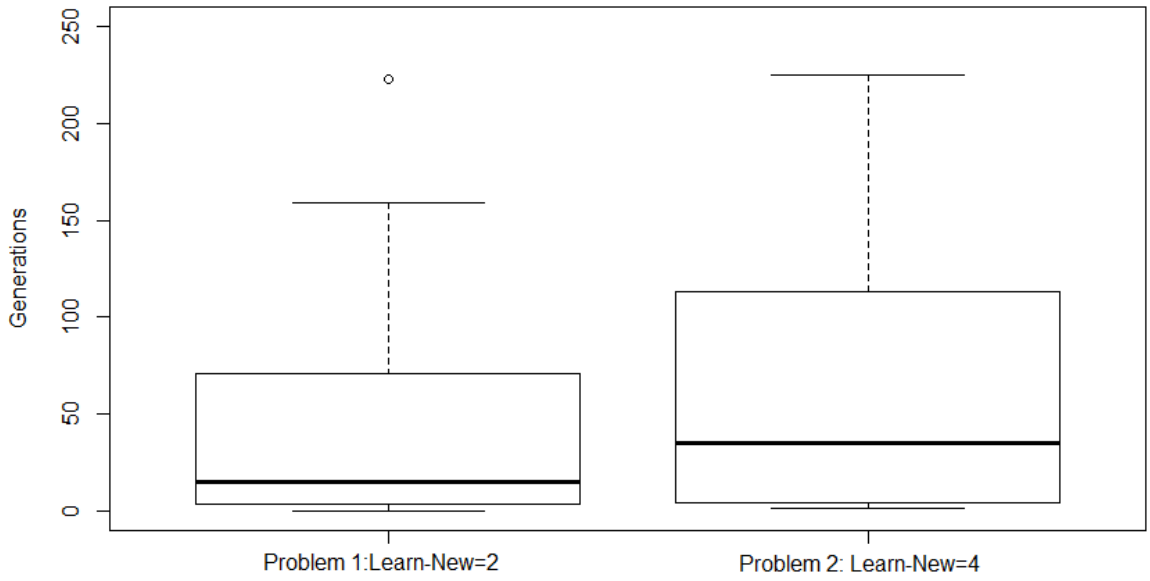

Figure 7.9: Plot two different tries of the experiment. This plot shows how long each experiment spends to find the optimal solution as generation numbers. The problem 1 shows fewer generations number compared to the problem 2. The learn-New relationship is different in each one. The learn-new relationship was 2 in the problem 1, and the Learn-New relationship was 4 in the problem 2. The GA operation is two point crossover.

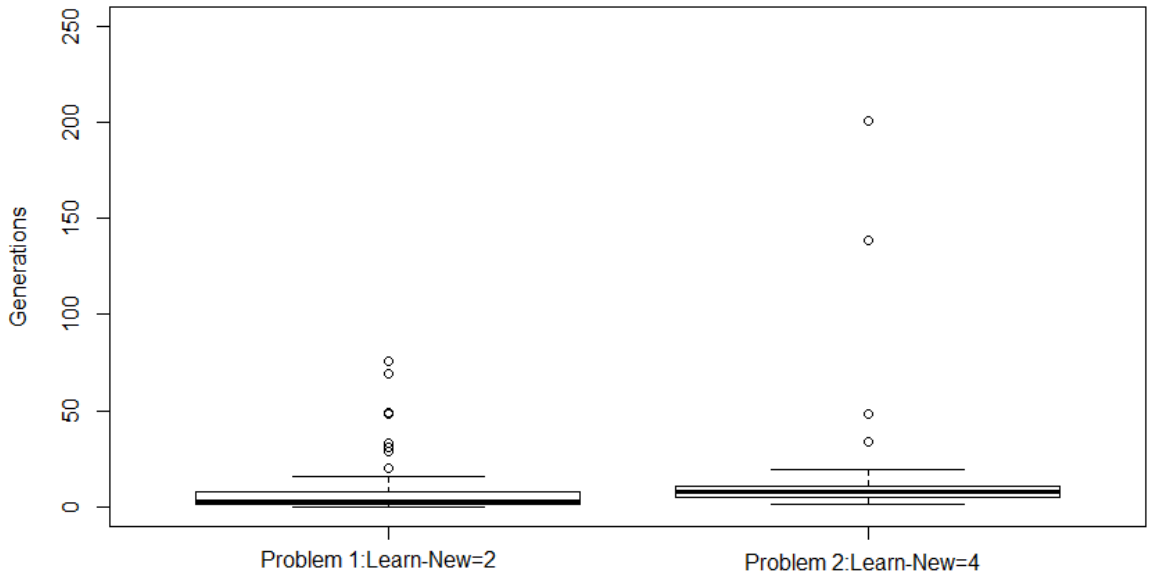

Figure 7.10: Plot two different tries of the experiment. This plot shows how long each experiment spends to find the optimal solution as generation numbers. Both of the experiments were close to each other as we use the uniform cross over in the GA. The learn-New relationship is different in each one. The learn-New relationship was 2 in the problem 1, and the Learn-New relationship was 4 in the problem 2.

Figures 7.9 and 7.10 show the hitchhiking situation. We used two point cross-over for the result in Figure 7.9, we used a one-point crossover for the result in Figure 7.10. This is why there is a difference between Figure 7.9 and 7.10.

As seen in Figures 7.9 and 7.10 and described in Table 7.1 the relationship is an important part that must discover before starting the TL process.

7.6 Conclusions

In conclusion, we have run three different experiments, and these experiments have a number of parts. We have shown the improvement of Transfer Learning in Genetic Algorithms. It is possible to transfer the knowledge between two different problems, in all of our experiments the source problems S and the target problem T were different than each other.

We showed in Experiment 7.5.1 that there exist problems S and T such that we can improve the solution of some problem T given the solution of problem S. We showed in Experiment 7.5.2 parts (A & B) that transferring the best individual was better than transferring all source

final population for the pairs of problems tried. We also showed in the same experiment that providing random noise to increase diversity in the populations was shown to be a hindrance for the pairs of problems tried. Discovering the relationship is an important part of the Transfer Learning Process. In Table 7.1 and Experiment 7.5.3, we showed a way to measure the relationship between S and T problems.

The main advantages of our work included fast adapting, and reducing the search time of the target problem task of finding the best solution. Transfer Learning can overcome some machine learning difficulties and GA, for example, start the learning process from scratch, and the absence of labeled trained data. According to the experiments that we performed and to the results in this paper, TL can be a tool that speeds up the learning process of GA and other machine learning algorithms.

Regarding to the three questions we stated previously in the introduction section, we can answer them as the following:

- **A1:** Yes, sharing of information or knowledge can help target problem be solved more quickly. As it appeared in the figures above, TL experiments took less time to solve a problem, 60% less generations. It solves the problem quickly and it provides a good solution. Sharing knowledge will perform better when there is a close relationship between S and T problems as mentioned above see (Section 7.5.1).

- **A2:** From Experiment 7.5.2 parts (A & B), The important part of the information from the previous population needed is the best individual. The best individual is the most important part of the knowledge to be transferred as it is showing in Figure 7.8.

- **A3:** The relations between the source S problem and the target T problem are an important part of the Transfer Learning Process. Also, we have discussed and shown one way of how relationships can be discovered in the Section 7.5.3, and Table 7.1.

Acknowledgments

This study was supported by the Saudi Arabian Cultural Mission (SACM) and Al Baha University, Al Baha, Kingdom of Saudi Arabia. I would like to pay my special regards to my supervisor professor Robert B. Heckendorn. His great advice for this study proved monumental towards the success of this study.

Chapter 8

Partial Knowledge Transfer Between The Source Problem and The Target Problem in Genetic Algorithms

This chapter is exactly as submitted at the ICECA 2022: 16. International Conference on Evolutionary Computation and Applications. It addressed the fifth objective, which is "Transferring partial knowledge from the source problem (S) to the target problem (T)". This chapter studies how to transfer partial knowledge from the S problem to the T problem. The results of evaluating this study showed that knowing something about how the problem was solved is better than starting from scratch. This paper answered the following questions:

Q.1 : Answer a concern of, does transferring partial knowledge from the source problem to the target problem help the target solver find the solution easily or not.

Q.2 : Study the influence of transferring partial knowledge of the source problem information to the target problem.

8.1 Summary

To study how the partial knowledge transfer may affect the Genetic Algorithm (GA) performance, we model the Transfer Learning (TL) process using GA as the model solver. The objective of the TL is to transfer the knowledge from one problem to another related problem. This process imitates how humans think in their daily life. In this paper, we proposed to study a case where the knowledge transferred from the S problem has less information than what the T problem needs. We sampled the transferred population using different strategies of TL. The results showed transfer part of the knowledge is helpful and speeds the GA process of finding a solution to the problem.

8.2 Introduction

Transfer Learning (TL) is the process of transferring the solution of one problem to help find the solution to another problem. It is one strategy of Machine Learning (ML) that aims to speed the process of finding solutions to related problems. TL can be thought of as a rich source of providing trained data to ML problems.

TL interacts with two problems. The first problem is the old one which we have data about. This problem is called the source problem, and it is symbolized with the letter (S). The second problem is the new problem, the problem that we are interested in solving more easily. This problem is called the target problem, and it is symbolized with the letter (T). The final solution that is chosen to transfer is called the transferred population [53].

The objective of the TL strategy is to transfer knowledge that has already been learned to the S problem. This behavior aims to reduce the effort and time that may be used to find the solution to the T problem if the target solver has started from scratch. The power of TL depends on how the S problem is related to the T problem. For example, if the S and T problems are related to each other or if they are from the same domain, TL may improve the performance of finding solutions to the T problem. On the other hand, if the S problem is not related to the T problem or if they are not from the same domain, the TL may reduce the performance and slow the process of finding the solution to the T problem. This situation is referred to as the negative transfer [53] [15].

Our goal is to study how efficiently to transfer knowledge between S and T problems if the S data is not enough or has partial information to help solve the T problem. The time and effort that may be used to solve the T problem from scratch may be better than transferring the partial information that the S solution has. Also, in a real-life scenario, it is hard to provide and find the exact data other problems need.

To address our thought, we model the TL process by employing the Genetic Algorithm (GA) as the model solver [21]. GA provides a solution to the problem by trying many candidate solutions and choosing the best one among all candidates. This process may take a lot of generations and iterations based on how hard the problem is. Our model is constructed to be tuned easily; however, we can control the difficulties and similarities of the S and T problems. Also, we can modify how many bits of an individual's problems need to be solved. Our model starts by solving the S problem. For this study, the requirements for solving the S problem are

less than the requirements for solving the T problem (see Section 8.4). After the GA finds the solution to the S problem, we construct the transferred population using the final S solution and different TL strategies, and transfer the transferred population to the T problem (see Section 8.5). The model uses the transferred population as the starting point to solve the T problem. The model counts how many generations the GA solver takes to find the final solution to the T problem.
We restricted ourselves to answering the following questions:

- **Q1.** Does transferring partial information of the S problem to the T problem help the T problem find the solution?
- **Q2.** Does information missing from the S problem have a major effect on the T problem performance compared to solving the T problem from scratch?

TL imitates how humans think. Humans usually consider their experience as a rich source they refer to when they face difficulties in their life. They try to find similar situations they have faced before and they use the lessons to solve the new problems. Also, sometimes humans ask other family members and friends about how to solve a problem. The question to ask is, what if the founded information is not enough to solve the problem?. Or what if a person found information that can solve half of the problem?. Will a person use it or not? A wise practice to this dilemma is to use what a person has and based on this information solve the whole problem. In ML and TL fields, we have data about how the problem is solved, and we transfer the knowledge to a related problem.

This paper is organized as follows: Section 8.3 is the background of other related studies, Section 8.4 is the method that we used in this paper, Section 8.5 is the experiment that we performed and the results. Section 8.6 is the discussion and the conclusion followed by acknowledgment.

8.3 Background

Shi and Sha [59] proposed a method for transfer knowledge for unsupervised learning approach. Their assumption was the data between the source and the target domains are similarly distributed. Their method learns the latent feature space between the source domain and the target domain also built the final classifier in one step. The authors tested their work using ob-

ject recognition data and sentiment analysis of Amazon products reviews. The results showed that their work has a significant improvement compared to other existing methods.

Xia and et al. [76] studied sentiment classification for Natural Language Processing (NLP). They implemented TL strategies to help improve the sentiment analysis for NLP tasks. They proposed a Feature Ensemble plus Sample Selection (SS-FE) algorithm for TL in NLP. SS-FE algorithm first, selects a subset of the source problem as the selected sample. Then, the algorithm used the selected data for labeling and transferring them to the target domain. The authors evaluated their algorithm using the reviewer's review from Amazon products. the results demonstrated this method improve the sentiment analysis compared to other methods.

Tommasi et al. [67] performed a study on objective recognition. They proposed a method based on the Least Square Support Vector Machine strategy that distinguishes the best data to model the model classifier among all data. Their method can learn the classifier from a small amount of data compared to other methods. The authors evaluated their method using the Caltech-256 data set. The results showed this method outperforms other methods. The authors claimed that this method reduces the effect of negative transfer.

Duan et al. [18] implemented TL with computer vision. They proposed the Domain Selection Machine (DSM) algorithm for event recognition. They transferred knowledge from (static) labeled pictures of the internet to indicate popular video events. DSM algorithm uses SIFT feature and space-time (ST) feature for data representations while other methods use one feature. They claimed the DSM algorithm can indicate the most relevant source domain. The result of experiments with three different data sets, showed the proposed algorithm significantly outperformed other existing methods.

8.4 Methods

Our model uses GA as the model solver for both problems (S and T). Our model gives us more access and control over the problems. The GA was chosen as the model's solver because it allows us to visualize the information the model knows about how a problem is solved, what information does the model know about the problem, and the effort the model spent to find the solution. This information can be found as the final solution to the source problem. The problem difficulties can be tuned easily by changing the fitness of the problem. The individual representation of a problem can easily be fixed and maintained. The transferred population consists of all information learned about how the source problem is solved. The transferred

population transfers to the target problem to help find the solution to the target problem.

For this study, we looked at cases where the individual representation of the S and the T problems is the same but the requirements for solving the S problem are different than the requirements for solving the T problem. In fact, the source requirements for solving the problem are less than the requirements for solving the T problem. This behavior allows us to study how to efficiently to transfer information that has some knowledge that is needed by the T problem. The rest of the T problem must be solved by the T solver (for more information see Figure 8.1).

Specifically, we used a source problem that must be solved and transfer the knowledge to the target problem. But the solution requirements for solving the source problem are less than the requirements for solving the target problem. Therefore, the target solver must solve the missing knowledge (chromosomes) between the S and the T problems in order to find a solution to the problem. In addition to this case, we studied cases that require the T solver to add new elements of knowledge to solve the problem.

Our model starts by initializing a random population. The S solver uses the random population to solve the S problem; after the solver finds the solution to the S problem, the final population transfers to the T problem. The T solver uses the transferred population as the starting point for solving the T problem. The model counts how many generations the T solver took to find the solution to the problem.

For this study, we choose to use generational GA as the model solver. Each individual consists of 40 bits, the GA works to find a solution to each problem using these bits. The individual who meets the fitness function requirement will be selected as the solution to the problem. Table 8.1 describes more details of the GA parameters.

Table 8.1: Generational Genetic Algorithm Parameters

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	100
Mutation rate (per bit)	0.1
Crossover rate	0.05
Type of crossover	Uniform crossover
Tournament Size	3

The fitness function evaluates how close an individual is to the desired solution. Each individual consists of 10 subfunctions and each subfunction has four bits. The length of each individual is 40 bits long. Our fitness function tries to maximize the solution to find the best

answer. We use the following function (7.2a) to find the fitness value of each individual. The function is defined over n bit string as follows:

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1]) \tag{7.2a}$$

where $m = 10$, $s = 4$, and $n = 40$, and $a_i \in \{0, 1\}$. The g_i is individual subfunctions. We have m subfunctions and these subfunctions are not overlapped. These subfunctions are defined over s bit subsection of the individual x. The $n = m * s$, $a_i \in \mathbb{R}$, and $x[a, b]$ is the bit position in the individual string x.

Individual's subfunctions must be solved in order to find the fitness value. It is not necessary to solve all subfunctions of an individual, just the necessary ones. For example, each individual has 10 subfunctions and these subfunctions consist of numbers of bits long. if $a_i = 1$, then the corresponding subfunction g_i needs to be solved and actually participate in the final solution. On the other hand, if the $a_i = 0$, then the corresponding subfunction g_i is not important and not needed in the final solution.

We use the Deceptive Function (Equation 7.2b) to solve these subfunctions. This function is a difficult, misleading type of function. The optimal solution is when all bits of a subfunction are equal to 1.

$$g(b) = \begin{cases} s & \text{bc}(b) = s \\ s - 1 - \text{bc}(b) & \text{otherwise} \end{cases} \tag{7.2b}$$

where $bc(b) \in \mathcal{B}^s$ is the bit counter function, $bc(b)$ is defined as bit $= 1$ in the b subsuction. The best answer is when all b bits are equal to 1.

For this study, the individual chromosome length is 40 bits for the source and the target population. But the requirement for the subfunction is different between the source and the target population, as follows: the optimal solution of the source's individual subfunction is when the total sum is equal to three. The order of bit that is equal to one does not matter for the individual's source subfunctions. On the other hand, the optimal solution of the target's individual subfunction is when all bits are equal to one and the total sum of the subfunction is equal to four. This way we can guarantee to transfer partial knowledge of the information from the source final population that is required by the target solver to use and to find a solution to the problem. Figure 8.1 can illustrate our thought more easily.

Figure 8.1 represents two individuals. The **S** represents the best individual from the S

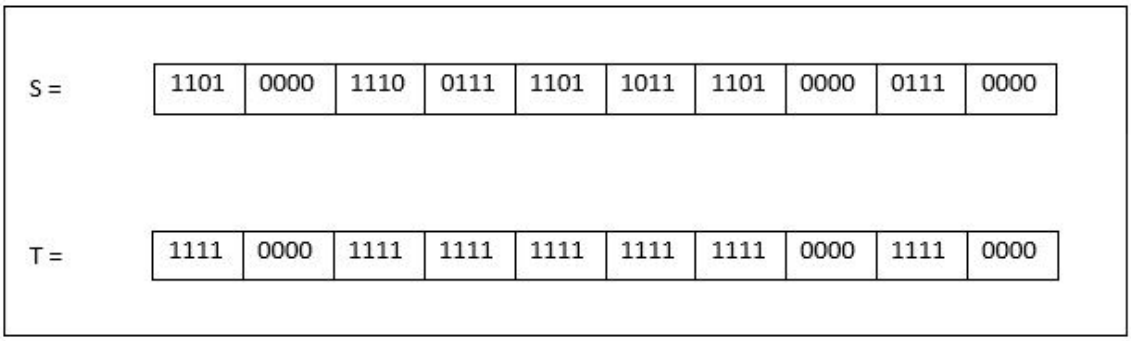

Figure 8.1: This diagram represents the source and target individual's genes. Each individual contains ten subfunctions and each subfunction contains four bits. The bit is either 0 or 1. **S** represents the best individual from the source population, where the source problem requests each subfunction to have three bits equal to one. **T** represents what the best solution of the T problem looks like. The target problem requests each subfunction to have all four bits equal to one in order for a specific subfunction to be included in the target final solution.

final solution. The **T** represents how the best individual of the T problem must look after the T solver finds the solution to the T problem. Both individuals have 10 subfunctions. As we discussed before, not all subfunctions need to be solved. In this Figure, if a subfunction contains three ones in its string, that means this subfunction contributes to the final solution of the S and T problems. We aim to transfer these subfunctions to the T problem. The T solver must solve the missing part of the transferred population to find a solution to the T problem.

We create two different experiments, to answer our questions in 9.2 Section.

The first experiment has one static source problem and four different target problems. These four target problems discover several approaches as follows: The first target problem (7.2ca) is identical to the S problem. But we must mention that the source's individual subfunctions requirements are different than the target's individual requirements. We expect that the T solver must work hard to find the missing information to have the right solution to the T problem. The second target problem (7.2cb) is different than the S problem by one element of knowledge. The third T problem (7.2cc) is different than the S problem by four elements of knowledge. The fourth T problem (7.2cd) is different than the S problem by eight elements of knowledge. In all approaches, the T solver must solve the T problem by solving the missing part of the transferred information from the S final population and adding the new elements of knowledge to the target final solution.

The following is the source problem:

$$\vec{a_s} = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0).$$

The following are the target problems (7.2ca, 7.2cb, 7.2cc, and 7.2cd):

$$\vec{a_{t1}} = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2ca}$$

$$\vec{a_{t2}} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2cb}$$

$$\vec{a_{t3}} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0) \tag{7.2cc}$$

$$\vec{a_{t4}} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1) \tag{7.2cd}$$

The second experiment has one source problem (7.2da) that transfers information to an identical target problem (7.2db) . The approach of this problem is different from the previous one. The S and the T problems are identical. The main approach in this problem is to find out if it is possible to transfer data from S to T problems where the S data is missing some information that is required for the T problem. We expected the T solver would work hard to find the missing information in order to find solution to the target problem. The S and the T problems are provided below:

$$\vec{a_s} = (1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2da}$$

$$\vec{a_{t1}} = (1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2db}$$

We hypothesize that transferring partial knowledge from the source problem to the target problem will result in speeding up the process of finding the solution to the target problem. We also hypothesize that transferring partial information to the target solver is better than starting the solving process from scratch. These two hypotheses are related to the two questions that we listed in the Introduction.

8.5 Experiments

We began by solving the S problem using a random population. We copied the random population and transferred it to the T solver as the Control-Unit (see Figure 8.2). The model used the random population and the $GA(S)$ to solve the S problem. We used the final solution of the S problem to construct the transferred population. We sampled the transferred population using five different strategies for constructing the transferred population. Then these sampled populations transferred to the T solver to find a solution to the T problem. The target solver also use the $GA(T)$ and the transferred population to solve the T problem. The model counted

how many generations it took to find the final solution to the T problem. For statistical and analysis purposes, the model ran 50 times.

The sampled strategies of the transferred population in our experiments are labeled as (100%-Top, 100%-Best, 30%-Top, 30%-Best, and EN-pop). Also, we created a copy of the first initial population for statistic purposes labeled as the "Control-Unit" (for more details see Figure 8.2).

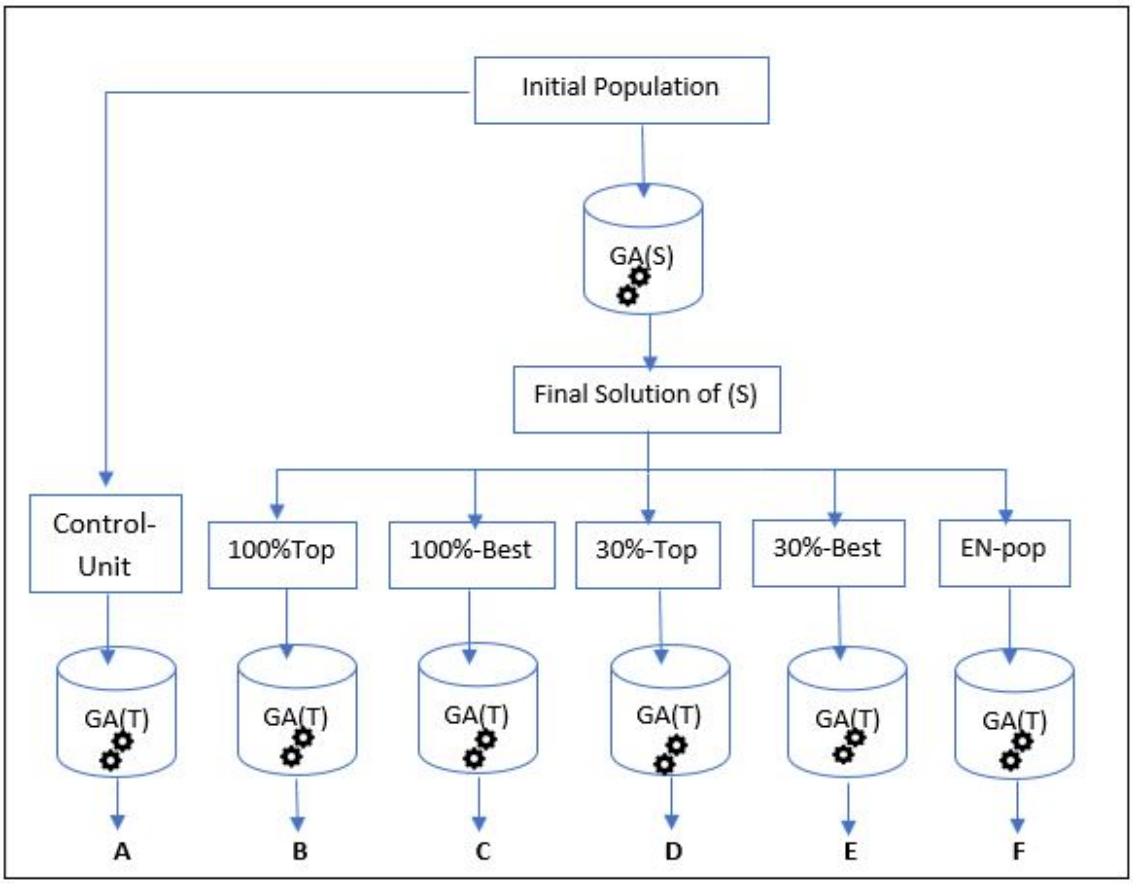

Figure 8.2: This diagram represents the performed experiment. A randomly initialized population is generated and transferred to the source solver $GA(S)$. A copy of this population is created for statistic purposes and transferred to the target solver $GA(T)$. After the $GA(S)$ finds a solution to the S problem, five different transferred population samples are created using the S final population. The sampled strategies are as follows: 100%-Top is an identical copy of the final solution of the S final solution; 100%-Best is a copy of the best individual 100% of the transferred population; 30%-Top is a copy of the top 30% of the S final population, and the remaining generated randomly; 30%-Best is a copy of the best individual of the S final population 30% of the transferred population size and the remaining generated randomly; and EN-pop is a sampling algorithm using entropy concept. All populations are transferred to the T solver to find a solution to the T problem, and the model counts how many generations the T solver took to find the solution.

The following is the explanation of the sample strategies of the transferred population we have used in this experiment:

- **100%-Top**: The whole final population of the S problem was copied into the transferred population.
- **100%-Best** The best individual of the S final population was copied 100% of the transferred population size.
- **30%-Top** The top 30% of the S final population was copied 30% of the transferred population size, and the remaining 70% was generated randomly.
- **30%-Best** The best individual of the S final population was copied 30% of the transferred population size, and the 70% remaining was generated randomly.
- **EN-pop** stands for entropy sampling, in which the transferred population is sampled using the entropy value of bit position of the S final population. If the entropy value of a bit (gene) of the S final population is less than or equal to 0.45 then the same (bit) or gene of the best individual of the S final population is copied to the transferred population, other ways generate random gene.

8.5.1 First Experiment

We studied how to transfer the knowledge from one static source problem to four different target problems.

8.5.1.1 First Target Problem

The first target problem (7.2ca) is a problem in which the S and the T problems are identical. The T solver does not have to add any new knowledge to solve the target problem. Instead of that, the T solver must find the missing information that transferred from the S final solution.

We ran the pairwise Mann-Whitney U test because the data is not normally distributed. This test is a free multiple comparison test of the five strategies that we performed. Table 8.2 shows the p-values of each strategy. Also, the last row shows the mean value of the number of generations for each strategy of the transferred population for solving the problem. Figure 8.3 shows how many generations the T solver spends to find the solution to the first target problem (7.2ca).

This experiment shows that transferring information from a problem that has less information than what the target problem requires is better than starting from scratch. The TL process

improves the GA performance. The 100%-Best strategy outperformed all other TL strategies. We can see this improvement illustrated in Figure 8.3.

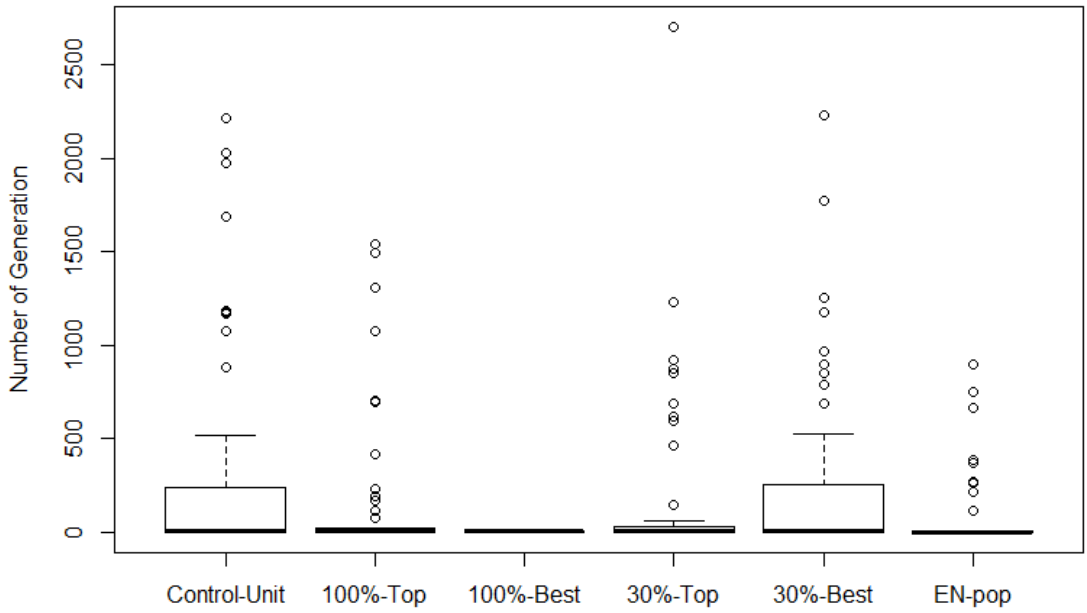

Figure 8.3: This diagram represents the number of generations the T solver takes to find the solution to the first target problem (7.2ca). The x-axis represents the transferred population sampled strategies. The y-axis represents number of generations. The T solver must find the missing genes between the S and the T problems. In this case, the S and T problems are identical. This problem ran for 50 iterations.

8.5.1.2 Second Target Problem

In the second target problem (7.2cb) the target solver must add must add one element of knowledge to find the solution to the second target problem and find the missing information form the transferred populations.

Figure 8.4 shows how many generations the T solver takes to find the solution to the second target problem (7.2cb). Table 8.3 shows the p-values of each strategies for solving the second target problem (7.2cb). Also, the last row shows the mean value of the number of generations for each strategy of the transferred population for solving the problem.

The TL improves the GA performance. In this case, this problem requires the T solver to find the missing information that transferred from the S final population and add one more

Table 8.2: This table is the pairwise Mann-Whitney U test for analyzing the first target problem of the first experiment 7.2ca. This table outputs a p-value. In our study, if the p-value is less than 0.01, that indicates the difference is significant between the strategies, and it is bolded. The last row shows the mean value of the number of generations to find the solution to the problem. The mean value for the EN-pop was 79.4 .

	30%-Best	30%-Top	100%-Best	100%-Top	Control-Unit
30%-Top	0.103	-	-	-	-
100%Best	0.341	0.122	-	-	-
100%-Top	0.487	0.243	0.994	-	-
Control-Unit	0.682	0.265	0.894	0.883	-
EN-pop	0.045	0.850	0.025	0.123	0.177
Mean	263.1	185.46	2.18	161.52	303.34

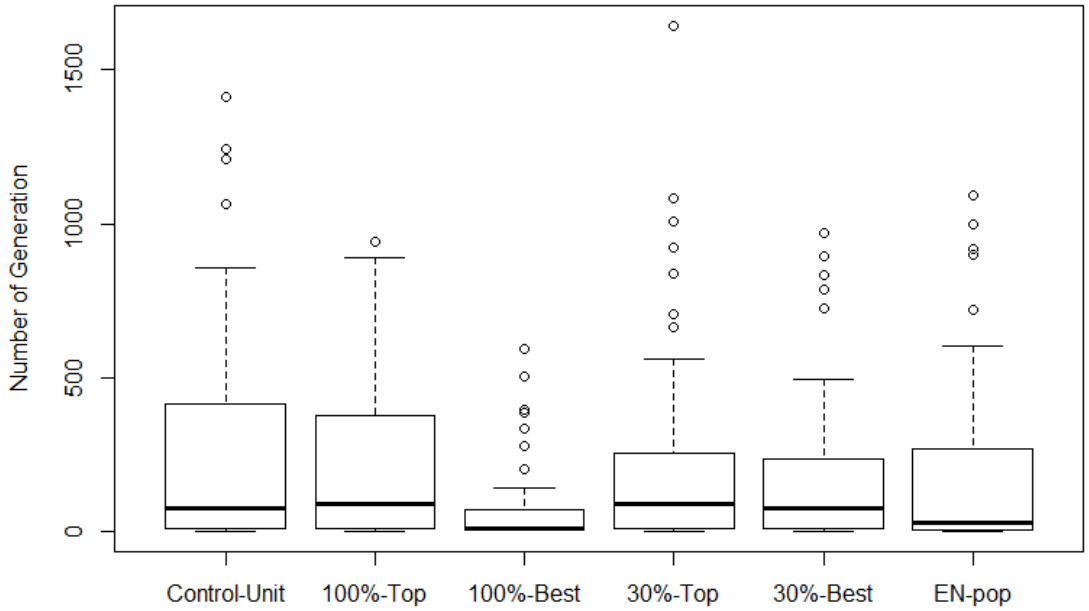

Figure 8.4: This diagram represents the number of generations the T solver takes to find the solution to the second target problem (7.2cb). The x-axis represents the strategies of the sampled transferred population. The y-axis represents the number of generations. In this problem, the T solver must add one element of knowledge and find the missing information from the transferred population. This problem ran for 50 iterations.

Table 8.3: This table is the pairwise Mann-Whitney U test for analyzing the second target problem of the first experiment 7.2cb. This table outputs a p-value. In our study, if the p-value is less than 0.01, that indicates the difference is significant between the strategies, and it is bolded . The last row shows the mean value of the number of generations to find the solution to the problem. The mean value for the EN-pop was 190.12.

	30%-Best	30%-Top	100%-Best	100%-Top	Control-Unit
30%-Top	0.855	-	-	-	-
100%Best	0.002	0.004	-	-	-
100%-Top	0.852	0.961	0.006	-	-
Control-Unit	0.923	0.961	0.008	0.923	-
EN-pop	0.215	0.202	0.335	0.194	0.203
Mean	148.88	236.4	80.46	205.84	262.68

extra knowledge to find a solution to the problem. Figure 8.4 illustrates that the TL strategies perform better than starting from scratch and the 100%-Best strategy outperforms all TL strategies.

8.5.1.3 Third Target Problem

The third target problem (7.2cc) required the T solver to add four elements of knowledge and find the missing information from the transferred population to find a solution to this problem.

Figure 8.5 shows how many generations the T solver takes to find the solution to the third target problem. Table 8.4 shows p-values for each strategies of the transferred population for solving this problem. Also, the last row shows the mean value of the number of generations for each strategy of the transferred population for solving the problem.

In this case, the T solver must add four elements of knowledge and find the missing information from the transferred population. The TL helps the GA solver to find the missing information and add the four new elements of knowledge to find a solution to the problem. Figure 8.5 illustrates the TL strategies help the GA performance and the 100%-Best strategy is the best strategy for sampling the transferred population.

Table 8.4: This table is the pairwise Mann-Whitney U test for analyzing the third target problem of the first experiment 7.2cc. This table outputs a p-value. In our study, if the p-value is less than 0.01 that, indicates the difference is significant between the strategies, and it is bolded . The last row shows the mean value of the number of generations to find the solution to the problem. The mean value for the EN-pop was 148.68.

	30%-Best	30%-Top	100%-Best	100%-Top	Control-Unit
30%-Top	0.600	-	-	-	-
100%Best	0.093	0.136	-	-	-
100%-Top	0.749	0.825	0.144	-	-
Control-Unit	0.860	0.257	0.022	0.634	-
EN-pop	0.506	0.767	0.202	0.667	0.250
Mean	166.34	152.44	130.7	162.34	173.9

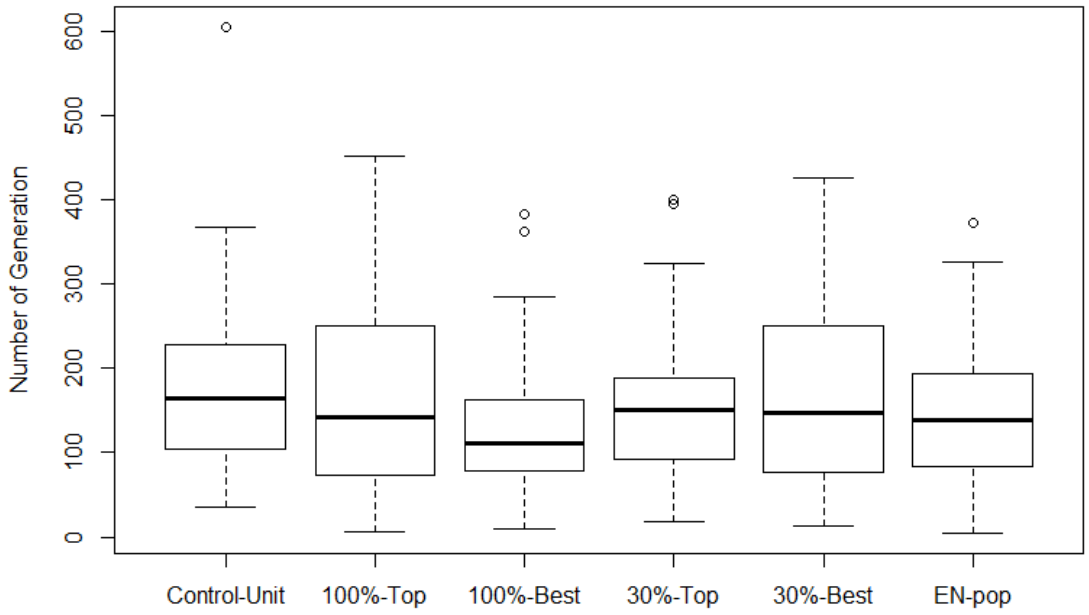

Figure 8.5: This diagram represents the number of generations the T solver takes to find the solution to the third target problem (7.2cc). The x-axis represents the strategies of the sampled transferred population. The y-axis represents the number of generations. In this case, the T solver must add four elements of knowledge and find the missing information from the transferred population. This problem ran for 50 iterations.

8.5.1.4 Fourth Target Problem

The fourth target problem (7.2cd) required the T solver to find the missing part of knowledge of the transferred populations, and add 8 elements of knowledge to find the solution to the fourth target problem.

Figure 8.6 shows how many generations the T solver takes to find the solution to the fourth target problem (7.2cd). Table 8.5 shows the p-values for each of the strategies of the transferred population for solving this problem. Also, the last row shows the mean value of the number of generations for each strategy of the transferred population for solving the problem.

This problem requires the T solver to add 8 elements of knowledge and to find the missing information to find the solution to this problem. Even though the T solver must add 8 elements of knowledge, the TL strategies help the GA performance to find a solution to the problem better than starting from scratch.

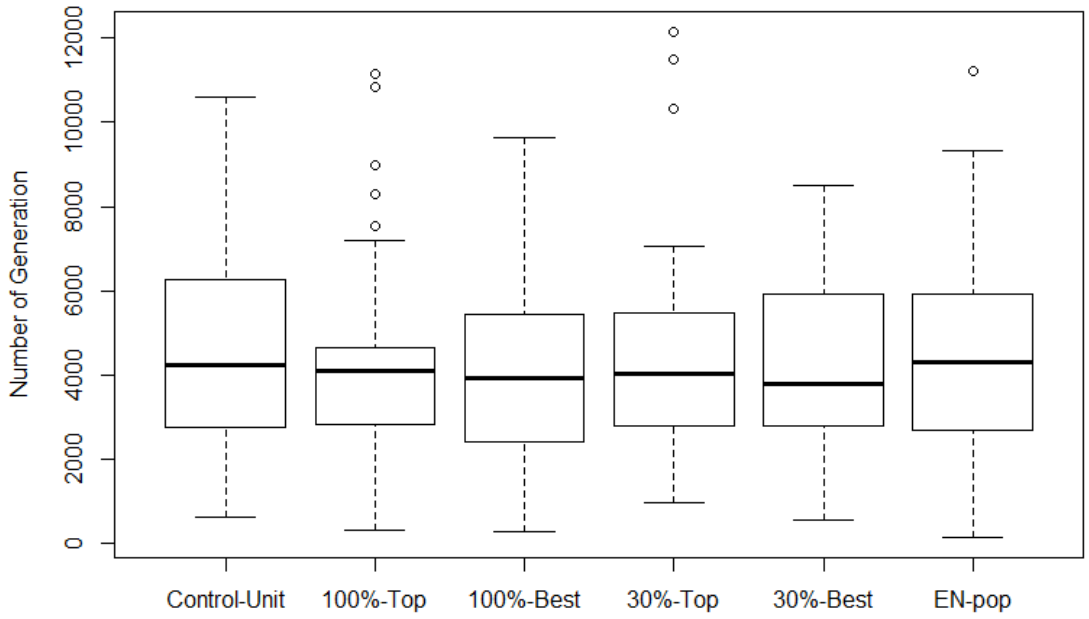

Figure 8.6: This diagram represents the number of generations the T solver takes to find the solution to the fourth target problem (7.2cd). The x-axis represents the strategies of the sampled transferred population. The y-axis represents the number of generations. In this case, the T solver must add eight elements of knowledge and find the missing information from the transferred population. This problem ran for 50 iterations.

Table 8.5: This table is the pairwise Mann-Whitney U test for analyzing the fourth target problem of the first experiment 7.2cd. This table outputs a p-value. In our study, if the p-value is less than 0.01, that indicates the difference is significant between the strategies, and it is bolded . The last row shows the mean value of the number of generations to find the solution to the problem. The mean value for the EN-pop was 4500.3.

	30%-Best	30%-Top	100%-Best	100%-Top	Control-Unit
30%-Top	0.98	-	-	-	-
100%Best	0.95	0.65	-	-	-
100%-Top	1.00	0.88	0.79	-	-
Control-Unit	0.62	0.70	0.49	0.63	-
EN-pop	0.72	0.73	0.48	0.66	0.90
Mean	4304.3	4431.24	4276.04	4357.1	4595.14

8.5.2 Second Experiment

This section is about the second experiment (7.2da). We studied how to transfer the knowledge from one source problem to an identical target problem. In this section, the target solver does not have to add knowledge to find the target solution, but it needs to find the missing information from the transferred populations. It is similar to the first target problem approach of the previous experiment (7.2ca).

8.5.2.1 Target Problem

This problem (7.2db) is a problem that does not required the T solver to add new knowledge to find the solution to the target problem, but it requires the T solver to fill or find the missing information from the transferred populations. Figure 8.7 shows how many generations the T solver takes to solve the (7.2db) problem. Table 8.5.2.1 shows the p-values for each of the strategies of the transferred population for solving this problem. Also, the last row shows the mean value of the number of generations for each strategy of the transferred population for solving the problem.

The TL strategies support the GA performance to find the missing parts of knowledge better than starting from scratch. Figure 8.7 and Table 8.5.2.1 illustrate the assistance of the TL strategies and GA performance. The p-value and the mean value of the 100%-Best strategy indicate that this strategy outperforms other TL strategies and starting from scratch.

8.6 Discussion

Our expectation that the T solver must work hard to find the solution to the missing information was not correct. Experiments 7.2ca and 7.2db showed that transferring information from the S final population whose information is less than what the T problem needs has improved the GA

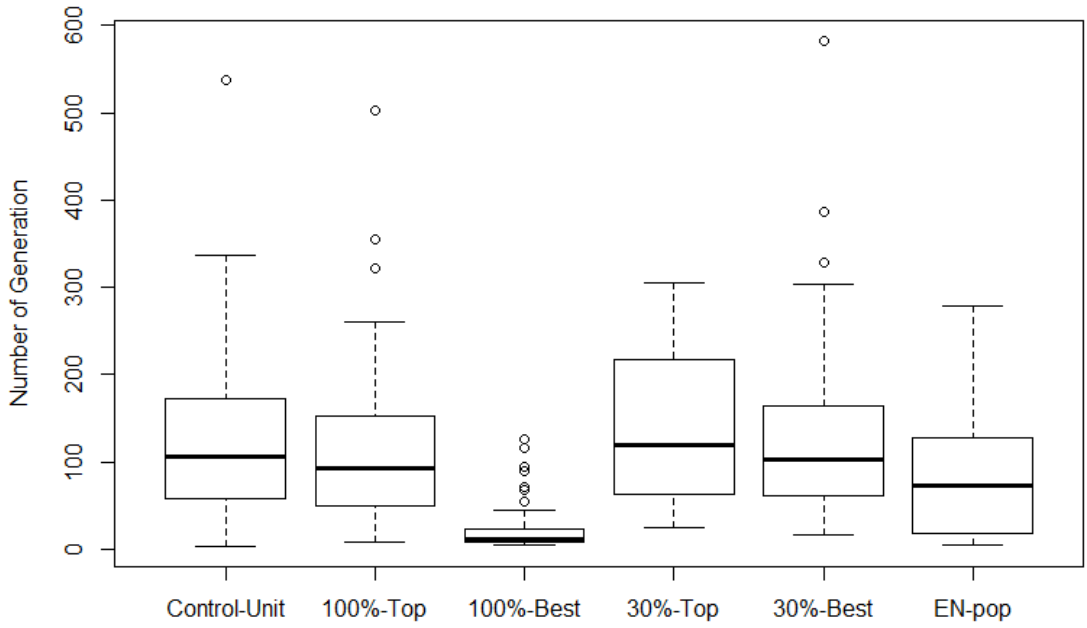

Figure 8.7: This diagram represents the number of generations the T solver takes to find the solution to the target problem (7.2db). The x-axis represents the strategies of the sampled transferred population. The y-axis represents the number of generations. The T solver must find the missing genes between the S and the T problems. In this case, the T solver does not have to add any element/s of knowledge to find the solution to the problem. This problem ran for 50 iterations.

Table 8.6: This table is the pairwise Mann-Whitney U test for analyzing the target problem of the second experiment 7.2db. This table outputs a p-value. In our study, if the p-value is less than 0.01, that indicates the difference is significant between the strategies. The last row shows the mean value of the number of generations to find the solution to the problem. The mean value for the EN-pop was 83.56.

	30%-Best	30%-Top	100%-Best	100%-Top	Control-Unit
30%-Top	0.442	-	-	-	-
100%Best	**4.914**	**1.514**	-	-	-
100%-Top	0.285	0.059	**3.111**	-	-
Control-Unit	0.664	0.214	**1.511**	0.758	-
EN-pop	0.003	0.0003	**5.07**	0.094	0.022
Mean	134.4	141.96	23.88	115.96	125.22

performance. The result showed the partial knowledge transfer guided the GA search process to find the missing information easily.

Using partial knowledge of how a problem was solved is better than starting from scratch. The TL strategies speed up and improve the performance of the model. Figure 8.3 through Figure 8.7 showed the 100%-Best strategy took fewer number generations compared to the other strategies of the TL.

In the cases where the S and the T problems are not identical, the solver has to add new elements of knowledge and find the missing part of the transferred gene. The TL strategies performed better than starting from scratch. The 100%-Best strategy took fewer generations compared to other strategies.

According to studies like [2] and studies such as [28], the diversity of the transferred population and the old knowledge plays a major role in the GA performance. In future research, we plan to target the subfunction's diversity and old knowledge and how this combination may affect the GA performance.

8.7 Conclusion

From the first and the second experiments we determined that, transferring partial knowledge to help solve another problem is beneficial and better than starting from scratch. The performance of employing the missing information in the process of finding the solution was more successful than dispensing the old knowledge. In other words, having something that may lead to finding the solution is better than nothing.

In this paper, we modeled the TL process. we have performed two experiments: the first one has four different T problems and approaches, and the second experiment has one T problem. Both experiments transfer knowledge from the S problem where the requirements for solving the S problem were less than the requirements for solving the T problem. The results showed transferring information that has missing data improved the GA performance of finding a solution to the T problem, in contrast to starting from nothing or from scratch.

Acknowledgment

This study is supported by Al Baha University, Al Baha, Saudi Arabia, and Saudi Arabian Cultural Mission (SACM). I would like to thanks my supervisor Professor. Terence Soule for his support and advice in this study.

Chapter 9

Transfer Knowledge From Multiple Source Problems to A Target Problem in Genetic Algorithm

This chapter is exactly as submitted at the ICSEC 2022: 16. International Conference on Swarm and Evolutionary Computation. It addressed the sixth objective, which is "Provide a study of transferring knowledge from multiple different source problems ($S_1, S_2, S_3, ...$) to the target problem (T). This study will allow me to understand the following tasks:". This chapter studies how to transfer knowledge from multiple S problems to the T problem. The results of evaluating this study showed that transferring knowledge from multiple sources to the target problem improve the GA performance and can cover different cases. This paper answered the following concerns:

Q.1 : Provide a case study of transferring knowledge from different source problems.

Q.2 : The possibility of transferring knowledge from multiple different source problems to a target problem.

Q.3 : Provide results of this study.

9.1 Summary

To study how to transfer knowledge from multiple source problems to the target problem, we modeled the Transfer Learning (TL) process using Genetic Algorithms as the model solver. TL is the process that aims to transfer learned data from one problem to another problem. The TL process aims to help Machine Learning (ML) algorithms find a solution to the problems. The Genetic Algorithms (GA) give researchers access to information that we have about how the old problem is solved. In this paper, we have five different source problems, and we transfer

the knowledge to the target problem. We studied different scenarios of the target problem. The results showed combined knowledge from multiple source problems improves the GA performance. Also, the process of combining knowledge from several problems results in promoting diversity of the transferred population.

9.2 Introduction

Transfer Learning (TL) is a process of transferring knowledge from one problem to another problem. The objective of TL is to speed up the process of finding the solution to the new problem. The mission of TL is to discover and transfer the trained data. This data may be adapted from multiple source problems.

TL is considered one of the Machine Learning (ML) techniques that help some ML algorithms find solutions to problems more easily. TL usually interacts with two problems: The first one is the source problem (S), and the other problem is the target problem (T). TL transfers the knowledge from the S problem to the T problem. In some cases, the S problem may not be presented, but the knowledge of solving the S problem is stored, and we can transfer it to the T problem [53].

The comprehensive process of TL mimics how humans think. For example, humans learn from their life how to adapt and solve problems or obstacles. They build on their experience from what they have faced and also what others have shared with them. Some people may listen to a lot of advice to understand and think about how to overcome his obstacles. Many people consider this behavior a wisdom behavior.

In this study, we adopt the following real-life situation. Students go to school and learn different subjects. They learn and practice what they have learned in class to pass and have good grades. For example, they apply what they learned in mathematics classes in physics classes. We would like to study how the possibility of transferring knowledge from several source problems to the target problem. Also, this study can be applied to different situations: for example, having two robots and each one has been trained to drive in two different areas. Robot A was trained to drive and discover the sand area. Robot B was trained to drive and discover the mountain area. Can we combine the knowledge from both robots in one robot that can drive and discover both areas?. This action will save us money and training time.

To address our goal, we modeled the TL process using Genetic Algorithm (GA) [21, 46]. The GA was used as the model solver for the S and the T problems. We created five different

source problems, and the model solved all of them first. Then we constructed the transferred population using the knowledge from all of these final solutions. We proposed using the Multi Bowl Transfer Population (MBTP) method to generate the transferred population (for more details, see Section 9.4). After that, the model transferred the constructed population to the target problems. The model solved the target problems using the constructed transferred population. To make our study cover different life situations, we tested four different target problems; each one describes different approach(for more information see Section 9.4).

Our study answers the following questions:

- **Q1.** Can we transfer knowledge from multiple source problems?
- **Q2.** How good is sampling the transferred population from multiple source problems?
- **Q3.** Can we solve a target problem that combined knowledge from two different source problems?

This paper is organized as follows: Section 9.3 explains the background of this study. Section 9.4 is the method we used in our study. The 9.5 is the experiment Section. 9.6 is the discussion Section followed by the conclusion and acknowledgments sections.

9.3 Background

Liu and Wang [38] implemented TL in Dynamic Multi-Objective Optimization Algorithms (DMO). They used TL to improve the initial population prediction for the target problem. They proposed an algorithm called TPS-DMOEA that contains the following steps:

1. Select the transferred population using the Population Prediction Strategy (PPS).
2. Modify the transferred population from the first step using Transfer Component Analysis (TCA).

The authors evaluated the proposed algorithm using ten different problems. The results showed the TPS-DMOEA algorithm overcame the existing methods of DMO.

Mendes et al. [16] implemented the TL to enhance CNN performance. They proposed a new method called Many Layer Transfer Learning Genetic Algorithm (MLTGA). They claimed their method can help the medical doctors to explore Pneumonia disease in early stages. The method built the Pneumonia's classifier model by transferring the well-trained layers from

previous CNN. The results showed the proposed algorithm was accurate with 2% higher than other GA methods.

Ardeh and et al. [4] studied the uncertain capacitated arc routing problem. This problem simulates an environment that has an undirected graph that connects tasks together, and the vehicles must serve these tasks. The best solution is to find the minimum cost that serves most tasks. The Genetic Programming Hyper Heuristic (GPHH) method is used to solve the previous problem. The authors enhanced the performance of the (GPHH) method using the TL. The proposed method discovers and removes the duplicated individuals from the transferred population. Also, the method maintains diversity that may be affected by the removing process. The experiment showed this approach overcomes state-of-the-art genetic programming with TL methods.

Chen and Liu [8] studied the Bi-Level Optimization Problems (BLOP). This problem is complex and is considered as nondeterministic polynomial (NP) type of problem. Typically, this problem deals with two levels of optimization. The low-level optimization controls the high-level optimization. Studies show this problem can be solved by using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The authors used the TL strategy to improve the performance of the (CMA-ES) strategy. Their strategy is composed of two steps. First, they restricted the search process to neighboring lower levels. Then they selected the transferred feature using the learning rate. The results showed the proposed algorithm improves the performance and efficiency compared to other CMA-ES enhanced methods.

9.4 Method

Our model employed the Genetic Algorithm (GA) as the solver unit. The GA allows us to know what information we have about solving a problem by analyzing the final solution of the problem. The GA generates a set of potential candidate solutions to the problem and searches among them to find the optimal's solution to the problem. The differences of the problems can be managed by controlling the fitness and the gene representations. The model blends the TL with GA by transferring the knowledge that has been learned to solve the S problem to the T problem. Our model counts how many generations the T solver took to find the solution to the T problem using the transferred population.

Our study looked at different perspectives in which the T problems require the T solver to search for different solutions. Our fitness function counted to learn new things. By changing the

fitness value of the T problem or the gene representation, we can discover different approaches.

For this study, we have developed five different source problems (Equations 7.2c) and four different target problems (Equations 7.2d). We consider cases where the target solver must learn new things, protect the knowledge the transferred population has, or forget some knowledge. We consider whether these cases required the solver to generate many generations to find the optimal solution. The number of generations reveals the efficiency of the transferred population. In this manner, we can study how effective our model is at transferring knowledge from multiple source problems.

The model starts by solving each one of the S problems using a random initialized population and stores the final solution of each problem separately. Then we construct the transferred population using the final solutions of all source problems that we stored (see Section 9.5). After that, the model transfers the transferred population to the T solver to find the solution to the T problem. The model counts how many generations the T solver took to find the final solution to the T problem.

In this study, we have generated four different T problems. Our model can study different cases and approaches. For example, what is the effect of having the system initiate a random T problem and ask the T solver to solve the problem? What is the effect of having a target problem requires combined knowledge from different S problems? What is the effect or the benefit of having a T problem simulate one of the S problems? What is the effect of having a T problem that requires partial data from different S problems? We considered if the T problems will require the T solver to generate many generations to find the solution to the T problem. We also considered if the transferred population is able to solve this type of problem easily or if the transferred knowledge is missing and not able to lead the T solver to solve the problem.

We have used the generational GA. The GA will deal with a population consisting of a fixed number of individuals. Each individual consists of 40 bits or genes long. For the selection operation of the GA, the tournament selection is enabled. In this selection, selected randomly individuals are selected and they have the right to produce the new generation. In this study, the tournament size is 3. The details of the GA arguments are specified in Table 9.1.

Fitness Function is a function that the GA used to evaluate each individual. This function yields a value, this value describes how close the evaluated individual is to solving the problem. For our study, we have used a fitness function that is defined on a number of bits or genes and divided them into many subfunctions. The following is the fitness function (Equation 7.2a).

Table 9.1: Parameters of The Generational Genetic Algorithm (GA)

Genetic Parameter	**Value**
GA Type	Generational
Chromosome length	40
Population size	100
Mutation rate (per bit)	0.1
Crossover rate	0.05
Type of crossover	Uniform crossover
Tournament Size	3

$$f(x) = \sum_{i=0}^{m} a_i \; g_i(x[i * s, \; (i+1) * s - 1]) \tag{7.2a}$$

where (x) is the individual. g_i is a subfunction, and it is defined over s bits subfunction. We have m subfunctions and s number of bits for each subfunction. The n is the total individual length. $n = m * s$ ($n = 10 * 4 = 40$). The a_i is [0,1] is the bit position of the x.

For our study, a_i value is fixed to [0,1]. We have chosen to deal with this type of problem because we want to show the importance of each crossbanding subfunction. For example, if the $a_i = 1$, that means the crossbanding subfunction is important and we have to solve it to achieve the solution to the problem. On the other hand, if $a_i = 0$, that means the crossbanding subfunction is not important and we do not have to find a solution to this subfunction. In other words, the only important subfunctions are the ones that crossband to 1.

We have 10 subfunctions that describe each individual. Each subfunction consists of four bits. Our model must solve each subfunction to evaluate each individual. To solve each subfunction we have implemented the deceptive function. This function is a type misleading subfunction. In general, this function shows it is improving as there is a zero in its argument, but the best solution is when all arguments or (subfunction's bits) are ones. The following is the deceptive function (Equation 7.2b).

$$g(b) = \begin{cases} s & \text{bc}(b) = s \\ s - 1 - \text{bc}(b) & \text{otherwise} \end{cases} \tag{7.2b}$$

where $bc(b)$ is the bit counter function. This function evaluates each subfunction. The best answer is when all bit of the subfunction are equal to 1.

For the feasibility of our study, we created five different source problems. We combined the transferred population from the final solution of these source problems. We developed four different target problems. The model transferred the combined population to the target solver.

The model dealt with each one of the T problems separately. Our model created four copies of the transferred population and solve each T problem individually. The following are the source problems:

$$\vec{a_{s1}} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2ca}$$

$$\vec{a_{s2}} = (0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0) \tag{7.2cb}$$

$$\vec{a_{s3}} = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1) \tag{7.2cc}$$

$$\vec{a_{s4}} = (0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2cd}$$

$$\vec{a_{s5}} = (0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0) \tag{7.2ce}$$

The following are the target problems. The first target problem (7.2da) has been chosen by the system.

$$\vec{a_{t1}} = (1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1) \tag{7.2da}$$

$$\vec{a_{t2}} = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1) \tag{7.2db}$$

$$\vec{a_{t3}} = (0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0) \tag{7.2dc}$$

$$\vec{a_{t4}} = (1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0) \tag{7.2dd}$$

As you can see, each one of the target problems discovers a different approach. For example:

- First T Problem 7.2da: GA(T-Random) random target problem. This problem will be initialized randomly by the system.
- Second T Problem 7.2db: GA(T-S1 & S3) combination of two source problems. This problem consists of a collection of the first S problem (7.2ca) and the third S problem (7.2cc).
- Third T problem 7.2dc: GA(T-S4) a simulation of the fourth S problem (7.2cd).
- Fouth T problem 7.2dd: GA(T-S1,S2,S5) a partial matching problem. This problem is a partial matching problem of the first, second, and fifth S problems (7.2ca, 7.2cb, and 7.2ce).

We hypothesize that we can transfer knowledge from multiple source problems to a target problem and this behavior will improve the GA performance. The TL can combine the

knowledge from different problems, and this combination will prevent knowledge from being losing. Also, this combination will add an amount of diversity to the transferred population. The diversity will increase the search space of the GA. The GA performance can be measured by the number of generations the GA must generate to find the solution to the problem.

9.5 Experiment

Our experiment looked at cases where the model samples the transferred population from multiple S problems and transfer them to the T solver. The target solver must use this population to find the solution to the T problem. We experimented with four different target problems. Each one of these problems cover a different approach.

In our experiment, we have five source problems denoted as ($S1, S2, S3, S4$, and $S5$) (see Figure 9.1). The source solver denoted as (GA) the solver used a randomly initialized population to solve each one of the source problems. The model solved and stored each problem individually. After the model finished solving all source problems, the model constructed the transferred population using the final solutions of all source problems. Then the model used the control unit population, a copy of each source's final population, and the constructed transferred population (MBTP) to solve the target problems. Each one of the target problems was solved individually. The model counted how many generations the target solver took to find the solution to each one of the target problems. This experiment ran for 50 iterations. For comparison purposes, we used the first initialized population (control unit) the final solutions of the source problems and the (MBTP) population. We compared the difference between these strategies. The following diagram represents the steps that we used in our experiment:

Figure 9.1 shows experiment diagram. We have five different source problems and four different target problems. Our model solves these problems and stores the final solution of each problem individually. Then the transferred population is constructed using the MBTP method. After that, the model will use the constructed population which is then the transferred population to solve each one of the target problems. The model counts how many generations the target solver used to find the solution to each one of the target problems.

Multi Bowl Transferred Population (MBTP) is the transferred population method we constructed for this study. In this study, we are dealing with five different source problems and we want to combine the knowledge of solving all S problems. We used 20% of each final solution of the S problems as follows:

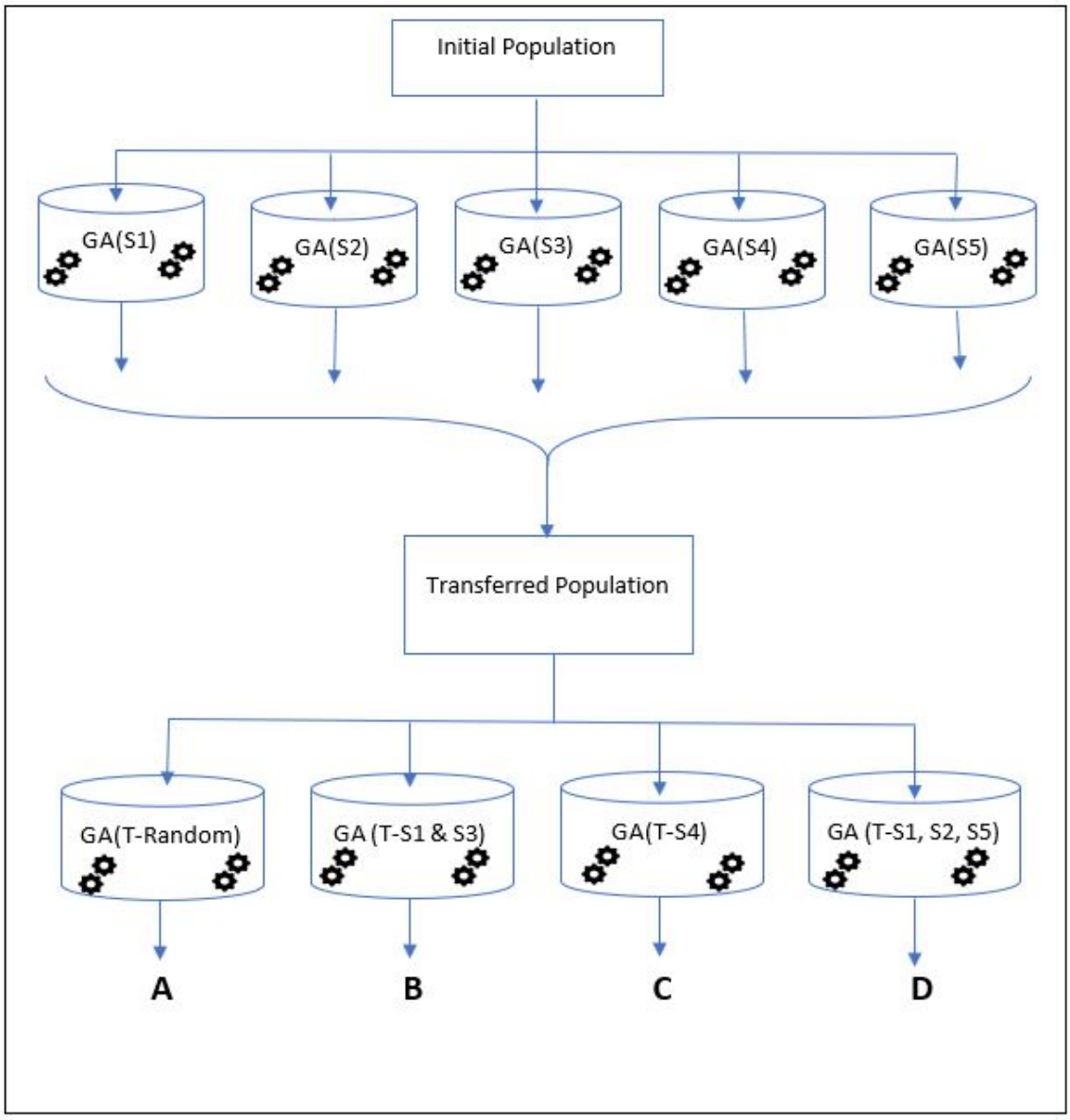

Figure 9.1: This figure represents the experiment. As it shows, we begin by solving the S problems. Each problem was solved individually, and the final solution was stored individually. The model solves each problem using random initialized population. After we solve all of the five problems, the model samples the transferred population using the MBTP method. After that we transfer the final population to the T problems. The T solver uses the transferred population to solve each T problem. We also copy the first initialized population four times (control unit). The model also uses these copies to solve each one of the target problems. The model counts how many generations the T solver took to find the solution to each one of the T problem. This experiment runs for 50 iterations.

- Top-10%: transferring the 10% of the top final solution of each S problem to the transferred population.
- Best-10% coping the best individual of each S problem 10% of the transferred population.

After we used the MBTP method to construct the transferred population, we transferred the final population to the T solver to solve the T problems.

9.5.1 First Target Problem

First target problem (7.2da) is the random problem. This problem was testing the random problem that was initialized by the system. The target solver must find a solution to this problem. Figure 9.2 represents how many generations the target solver used to solve this problem.

Figure 9.2 represents how many generations the target solver took to find a solution to the first target problem (7.2da). The fourth and MBTP populations show fewer number of generations compared to other populations. The control unit population represents the first random population. The target solver takes a large number of generations using the control unit population to solve this problem.

Table 9.2: This is the p-value of the pairwise Mann-Whitney U test for solving the first target problem 7.2da. In our study, if the p-value is less than 0.01, that indicates the difference is significant between each type of the population, and we bolded it. The last row shows the average value of the number of generations to find the solution to the problem. The mean value for the third-solution was 120.56.

	control-unit	fifth-solution	first-solution	fourth-solution	MBTP	second-solution
fifth-solution	0.928	-	-	-	-	-
first-solution	0.018	0.021	-	-	-	-
fourth-solution	0.0005	0.0007	0.156	-	-	-
MBTP	0.0004	0.0004	0.106	0.764	-	-
second-solution	0.156	0.194	0.273	0.018	0.011	-
third-solution	0.475	0.541	0.085	0.002	0.0009	0.404
Mean	141.34	143.04	99.08	87.7	73.68	109.1

Table 9.2 is the pairwise Mann-Whitney U test. This test shows the significance (p-value) of using each type of population to solve the problem. The last row of this table shows the average number of generations for each strategy.

9.5.2 Second Target Problem

Second target problem (7.2db) is the combination problem. This problem tests the ability of the model to solve a problem that combines knowledge from two different source problems.

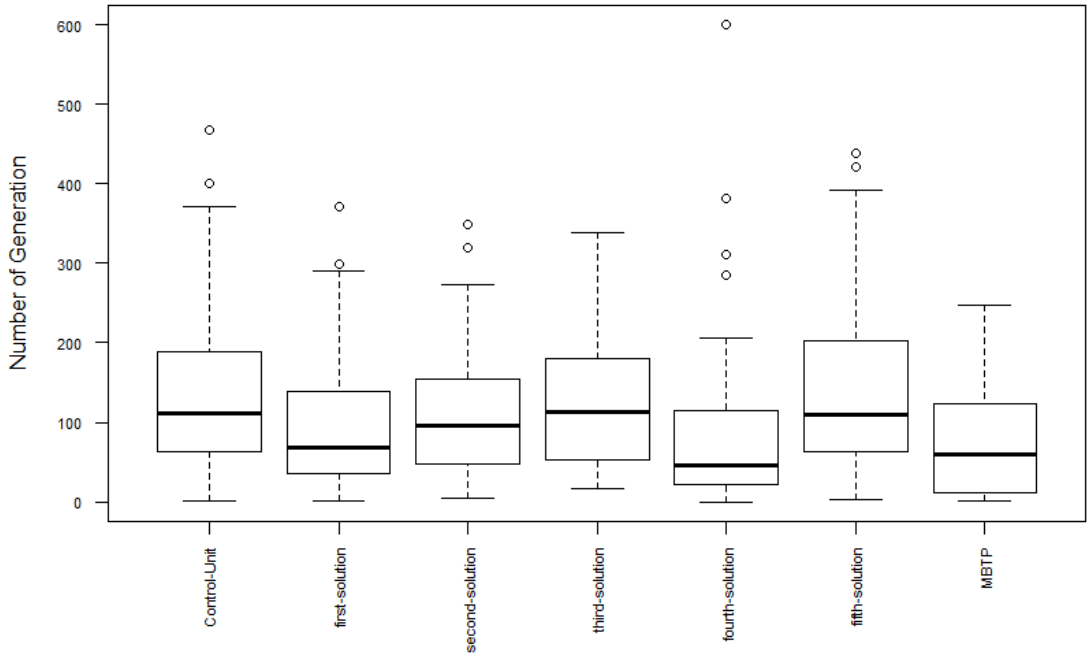

Figure 9.2: This graph represents how many generations the target solver took to solve the first target problem, and also shows seven types of populations. The target problem used in this graph was the random problem 7.2da. The target solver spends fewer number of generations using the fourth-population and the MBTP population. This experiment runs for 50 iterations.

This problem combines knowledge from S problem (7.2ca) and S problem (7.2cc). The target solver must use the transferred populations to find a solution to this problem.

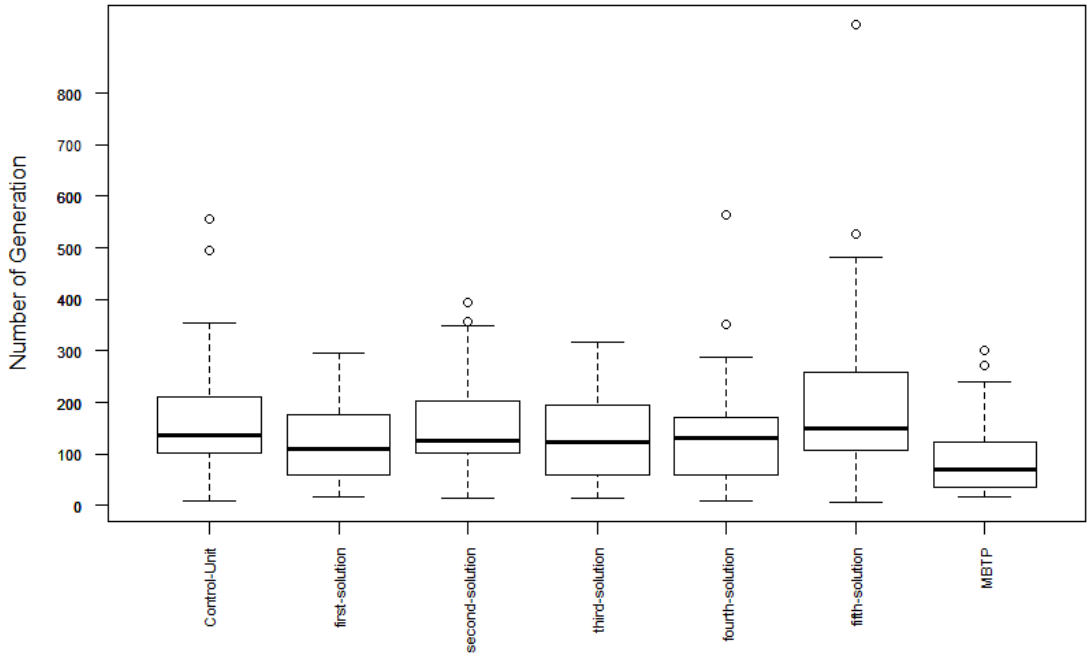

Figure 9.3: This is a boxplot diagram that represents how many generations the T solver spent to find the solution to the second target problem 7.2db. The second target problem is the combined problem, which combined knowledge from S problem 7.2ca and S problem 7.2cc. The x axis shows the type of population and the y axis shows the number of generations.

Figure 9.3 represents how many generations the T solver spends to find the solution to the problem. The MBTP population shows fewer number of generations compared to the other population. The final solution of the fifth problem spends a large number of generations.

Table 9.3: This table shows the pairwise Mann-Whitney U test result for solving the second target problem 7.2db. This table outputs a p-value. In this study, if the p-value is less than 0.01, that indicates the difference is significant between population types and we highlight this value. The last row shows the average value of the number of generations for solving the problem. The average value for the third-solution was 130.3.

	control-unit	fifth-solution	first-solution	fourth-solution	MBTP	second-solution
fifth-solution	0.612	-	-	-	-	-
first-solution	0.052	0.011	-	-	-	-
fourth-solution	0.059	0.034	0.890	-	-	-
MBTP	**7.406**	**1.406**	0.007	0.003	-	-
second-solution	0.475	0.244	0.165	0.339	**5.005**	-
third-solution	0.077	0.019	0.814	0.893	**0.003**	0.238
Mean	168.74	196	127.52	135.56	87.82	150.64

Table 9.3 shows the pairwise Mann-Whitney U test results of the number of generations. This test shows the p-value of each sampled population. The last row of this table shows the

average number of generations for each strategy.

9.5.3 Third Target Problem

Third target problem (7.2dc) is the simulation problem. This problem is exactly the same as the fourth source problem (7.2cd). The aim of this problem was to test if the knowledge stored in the transferred population solves the problem easily or if the model can improve the solution to this problem. The model must find the solution to this problem.

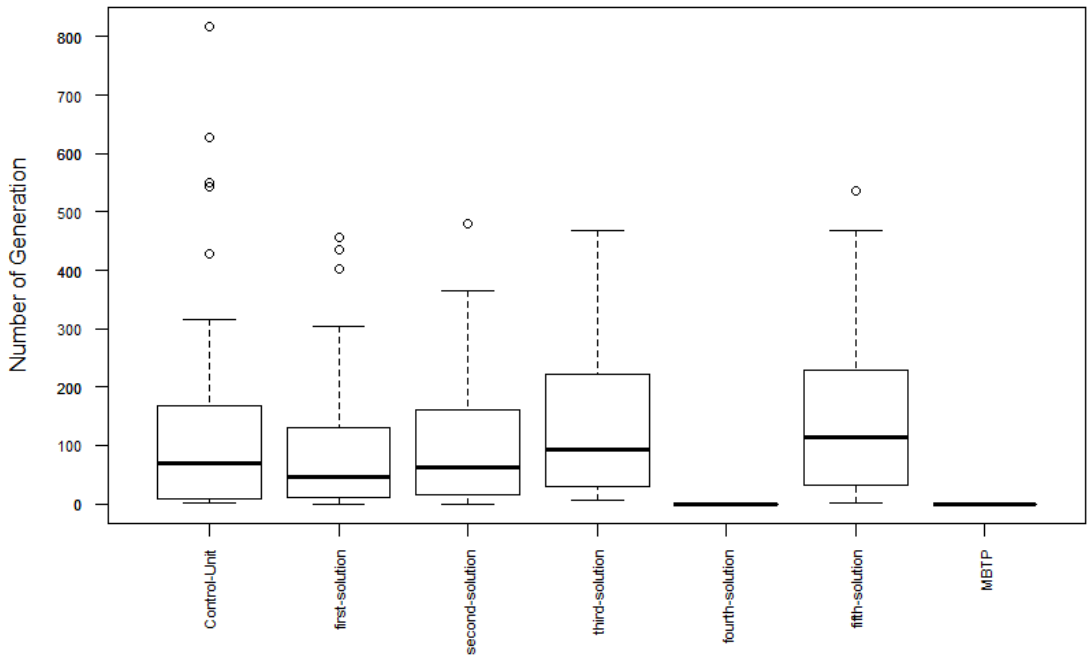

Figure 9.4: This diagram represents how many generations the T solver took to find the solution to the third target problem 7.2dc. The x axis represents each type of population, and the y axis represents the number of generations. The fourth-solution and MBTP populations show 0 generations, because these two populations have solutions to this problem.

Figure 9.4 represents how many generations the T solver takes to find a solution to the fourth T problem. This figure shows the target solver spends 0 generations using the final solution of the fourth source problem (7.2cd) and the MBTP populations. This is because these two populations already have the solution for this problem.

Table (9.4) is the pairwise Mann-Whitney U test. This test shows the significant p-values for each transferred population. The last row represents the average number of generations for each population.

Table 9.4: This table shows the pairwise Mann-Whitney U test results for the number of generations the T solver spent to solve the third target problem 7.2dc. This table outputs a p-value. In this study, if the p-value is less than 0.01, that indicates the difference is significant between the populations. The last row shows the average value of the number of generations for finding the solution to the third target problem. The average value for the third-solution was 132.2. The fourth-solution and MBTP populations have fewer value, this is because these two populations already have the solution.

	control-unit	fifth-solution	first-solution	fourth-solution	MBTP	second-solution
fifth-solution	0.107	-	-	-	-	-
first-solution	0.375	0.088	-	-	-	-
fourth-solution	**<216**	**<216**	**<216**	-	-	-
MBTP	**<216**	**<216**	**<216**	-	-	-
second-solution	0.902	0.057	0.415	**<216**	**<216**	-
third-solution	0.137	0.745	0.017	**<216**	**<216**	0.104
Mean	134.4	150.28	88.64	0	0	104

Table 9.5: This table shows the pairwise Mann-Whitney U test results, for solving the fourth target problem 7.2dd. This table outputs a p-value. In this study, if the p-value is less than 0.01, that indicates the difference is significant between each type of population. The last row shows the average value of the number of generations for solving the fourth target problem. The average value for the third-solution was 178.3.

	control-unit	fifth-solution	first-solution	fourth-solution	MBTP	second-solution
fifth-solution	0.108	-	-	-	-	-
first-solution	0.574	0.236	-	-	-	-
fourth-solution	0.330	0.318	0.801	-	-	-
MBTP	0.006	0.360	0.031	0.045	-	-
second-solution	0.055	0.955	0.201	0.323	0.491	-
third-solution	0.183	0.616	0.400	0.614	0.168	0.552
Mean	261.5	192.98	234.62	198.7	97.52	186.56

9.5.4 Fourth Target Problem

Fourth target problem (7.2dd) is is a partial matching problem. This problem has knowledge from source problems one, two, and five (7.2ca, 7.2cb, and 7.2ce). This problem aims to test the model ability to solve a problem that required knowledge from multiple different parts of the final solutions of multiple source problems. The target solver must use parts of the knowledge that is stored in the population to find a solution to this problem.

Figure (9.5) represents how many generations the T solver spends to find the solution to this problem. The MBTP population shows fewer number of generations compared to the other population type.

Table (9.5) is the pairwise Mann-Whitney U test. This table shows the p-values of the significance level of the number of generations for each population type. The lase row shows the average number of generations the T took to find the solution to the problem.

9.6 Discussion

This study showed transferring knowledge from multiple source problems is possible and may help algorithm designers to improve the GA performance. Knowledge from multiple sources

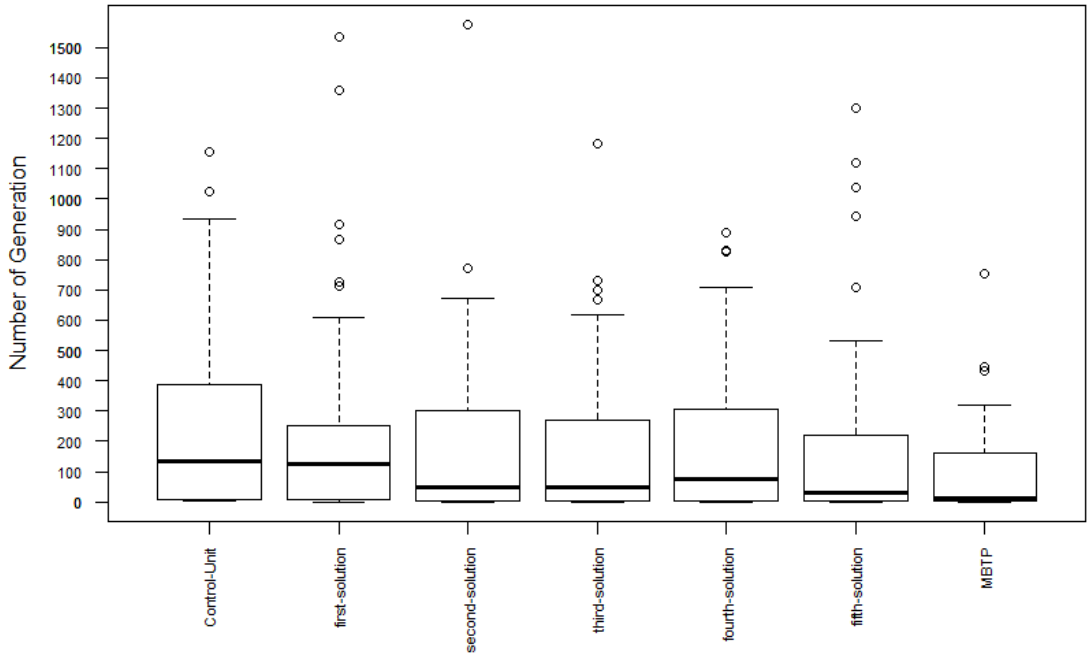

Figure 9.5: This figure represents the number of generations the T solver took to find a solution to the fourth target problem 7.2dd. This is the partial transfer problem. This problem has information from the first, second, and fifth target problems. The MBTP spends fewer generations compared to the other population types. This problem runs for 50 iterations.

can be combined together in one population and solve more advanced problems. For example, our study gathers knowledge from five different source problems and applied it to solve the target problem.

By analyzing the transferred population that contains the combined knowledge, we found the process of combining knowledge from multiple source problems added some diversity to the transferred population. Also, the combining process protected the old knowledge from loss. According to other studies such as [28] and [2], old knowledge and diversity are two important components that must be available in the transferred population.

The MBTP method constructed the transferred population using knowledge from final solutions of multiple source problems. This method follows the fashion of exploiting and exploring. The exploitation was enhanced by using the 10% copy of the best individual of each source problem, and the exploration was enhanced by transferring the top 10% of each source problem's final solution.

In this study, we have used diversity from the source problems that we had already used to

solve our problems. We transferred the top 10% of each final solution to the transferred population to enhance the population diversity. We feel this diversity did not cover all possibilities since it was used to solve other problems. In the future, we must investigate what happens if we use a totally random individual as population diversity.

9.7 Conclusion

We studied how to transfer knowledge from multiple source problems. We proposed the MBTP method, which samples the transferred population using knowledge and diversity from solutions of the solved source problems. We experimented with five different source problems. We constructed the transferred population and transferred it to the target solver to solve the target problems.

We studied four different approaches of target problems. These approaches cover some real-life scenarios. For example, if we have two robots that had been trained to drive in two different environments, we can combine their knowledge into one robot that can serve in both environments.

Transfer Learning can combine data from multiple sources in one population. Our proposed method may help the GA task to solve more advanced problems or at least protect the knowledge from loss. The experiment and results show gathering knowledge from multiple sources improves the GA performance compared to starting from scratch.

acknowledgment

This study is fully supported by Al Baha University, Al Baha, Saudi Arabia, and (SACM) Saudi Arabian Cultural Mission. I would like to thanks my supervisor Professor. Terence Soule for his support and advice in this study.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

This dissertation focused on understanding the requirements for successfully using Transfer Learning in Genetic Algorithms. We modeled the process of TL by employing a GA as the solver. We began the research by evaluating several strategies for discovering the best methods of transferring knowledge from a source problem to the target solver. We proposed an algorithm that samples the transferred population based on the source's final knowledge. We evaluated the effect of the transferred population content on the target solver, and we analyzed which and why some strategies of TL are more successful than others. We analyzed several factors that affect the GA process. Finally, we suggested how best to implement TL in GA and that the best approach depends on the similarity between the source and target problem. We studied the impact of transferring partial knowledge from the source problem to the target problem and how to sample and transfer knowledge from multiple source problems to a single target problem.

10.2 Future Work

The first section, 10.2.1, is about problem size and how the size of a problem would affect the performance of the model. The second section, 10.2.2, is about individual diversity and population diversity, and how they are different than each other. The third section, 10.2.3, is about multiple source problems and how to collect the knowledge from multiple source problems and transfer it to the target problem.

10.2.1 Impact of Problem Size

The size of a problem is a major factor in how easy it is to solve using ML. As this study did not cover the impact of problem size, I would like to perform a study of how the size of the

problem affects the model's performance.

In these experiments with the TL process using a GA as the model solver, each individual consists of 10 subfunctions, and each subfunction consists of four bits. Future research would include comparison studies between different problem sizes. For example, a problem that takes 2 bits as the subfunction size and a total of 10 subfunctions, a problem that takes 4 bits as the subfunction size and a total of 10 subfunctions, a problem that takes 6 bits as the subfunction size and a total of 10 subfunctions, a problem that takes 8 bits as the subfunction size and a total of 10 subfunctions, and a problem that takes 10 bits as the subfunction size and a total of 10 subfunctions.

This future research would examine how many generations it took to solve the source and the target problems as a funciton of the size and number of subfuntions. The ratio between the target problem number of generations and the source problem number of generations for each problem will be plotted to see the effect of the size on each problem.

10.2.2 Individual Diversity vs Population Diversity

This dissertation dealt with two types of diversity: population diversity and individual diversity. The effect of population diversity is different than the effect of individual diversity in terms of GA performance. An important avenue of future research is exploring individual diversity. When, how, and in what cases would using individual diversity be better rather than using population diversity? As discussed in the partial transfer knowledge Chapter 8, the result of transferring the final population of the source problem helps the GA model to find the solution more easily, although it has some missing information the target problem needs. The results show that individual diversity has a major impact in that case, demonstrating its importance in some problem types.

Chapter (5) of this dissertation proposed an algorithm that sampled the transferred population based on individual entropy values. It amy be possible to enhance the sampling process by adding random diversity to each individual of the transferred population. It would be valuable to study how this change would affect GA performance.

10.2.3 Multiple Sources

Chapter 9 dealt with transferring knowledge from multiple source problems. We combined the knowledge by constructing a multi-bowl population using the Multi Bowl Transferred Population algorithm. Our method combined the knowledge from all source problems using the same

amount of knowledge and diversity. This case would benefit from additional research because having many trained solutions from different sources in one machine is likely to help discover more situations, and speed up the learning process.

Chapter 9 maintained the population diversity by transferring 10% from each solution of each source problem. It would be interesting to study different cases where we generate a totally random population diversity and compare how this affects the GA performance instead of using diversity that is already present in the source population.

Chapter 9 constructed the transferred population in one step using the MBTL method. MBTL transferred 20% of each final solution of each source problem to the MBTL bowl (10% as old knowledge and 10% as diversity). It would be interesting to use a different technique, instead of constructing the transferred population in one step, I am suggesting having a Cumulative Transferred Population (CTP). The CTP transfers the final solution of the first source problem to help solve the second source problem and transfer the solution of the second source problem to help solve the solution of the third source problem etc. This process will repeat until all source problems are solved . After that, we will transfer the final solution of the last source problem to the target problem.

Bibliography

[1] *Transfer Learning in Genetic Algorithms*. PSRC, 2012.

[2] T. Alghamdi and R. B. Heckendorn. An evolutionary computation based model for testing transfer learning strategies. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1380–1389. IEEE, 2021.

[3] L. Alzubaidi, M. Al-Amidie, A. Al-Asadi, A. J. Humaidi, O. Al-Shamma, M. A. Fadhel, J. Zhang, J. Santamaría, and Y. Duan. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, 13(7):1590, 2021.

[4] M. A. Ardeh, Y. Mei, and M. Zhang. Surrogate-assisted genetic programming with diverse transfer for the uncertain capacitated arc routing problem. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 628–635. IEEE, 2021.

[5] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48, 2007.

[6] W. Banzhaf, R. Keller, and F. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

[7] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003.

[8] L. Chen and H.-L. Liu. Transfer learning based evolutionary algorithm for bi-level optimization problems. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1643–1647. IEEE, 2021.

[9] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu. Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545, 2007.

[10] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[11] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 200–207. ACM, 2008.

[12] C. Darwin. *On the Origin of Species: A Facsimile of the First Edition*. Harvard University Press, Cambridge, MA, USA, 2001.

[13] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[14] H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

[15] O. Day and T. M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):1–42, 2017.

[16] R. de Lima Mendes, A. H. da Silva Alves, M. de Souza Gomes, P. L. L. Bertarini, and L. R. do Amaral. Many layer transfer learning genetic algorithm (mltlga): a new evolutionary transfer learning approach applied to pneumonia classification. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 2476–2482. IEEE, 2021.

[17] T. T. H. Dinh, T. H. Chu, and Q. U. Nguyen. Transfer learning in genetic programming. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1145–1151. IEEE, 2015.

[18] L. Duan, D. Xu, and S.-F. Chang. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 1338–1345. IEEE, 2012.

[19] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Viena, Austria, 2003.

[20] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Spinger, second edition, 2015.

[21] A. E. Eiben, J. E. Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[22] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

[23] D. B. Fogel. What is evolutionary computation? *IEEE spectrum*, 37(2):26–32, 2000.

[24] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 283–291. ACM, 2008.

[25] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

[26] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing, Co., Reading, MA, 1989.

[27] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

[28] A. Gupta and Y.-S. Ong. Genetic transfer or population diversification? deciphering the secret ingredients of evolutionary multitask optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2016.

[29] R. B. Heckendorn. Embedded landscapes. *Evolutionary Computation*, 10(4):345–376, 2002.

[30] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

[31] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[32] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. ACL, 2007.

[33] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[34] A. Khalaf, E. Sejdic, and M. Akcakaya. Mutual information for transfer learning in ssvep hybrid eeg-ftcd brain-computer interfaces. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 941–944. IEEE, 2019.

[35] B. Koçer and A. Arslan. Genetic transfer learning. *Expert Systems with Applications*, 37(10):6997–7002, 2010.

[36] P. Larrañaga and J. A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2001.

[37] F. Li, S. J. Pan, O. Jin, Q. Yang, and X. Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 410–419, 2012.

[38] Z. Liu and H. Wang. Improved population prediction strategy for dynamic multi-objective optimization algorithms using transfer learning. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 103–110. IEEE, 2021.

[39] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic. *Measurement*, 167:108288, 2021.

[40] M. Long, J. Wang, G. Ding, D. Shen, and Q. Yang. Transfer learning with graph co-regularization. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1805–1818, 2013.

[41] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[42] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[43] M. Mahmud and S. Ray. Transfer learning using kolmogorov complexity: Basic theory and empirical evaluations. In *Advances in neural information processing systems*, pages 985–992, 2008.

[44] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Aaai*, volume 7, pages 608–614, 2007.

[45] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[46] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[47] B. Muller, H. Al-Sahaf, B. Xue, and M. Zhang. Transfer learning: a building block selection mechanism in genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 350–351, 2019.

[48] S. Nagae, S. Kawai, and H. Nobuhara. Transfer learning layer selection using genetic algorithm. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2020.

[49] R. M. Nelson, M. Kierczak, and Ö. Carlborg. Higher order interactions: detection of epistasis using machine learning and evolutionary computation. In *Genome-Wide Association Studies and Genomic Prediction*, pages 499–518. Springer, 2013.

[50] S. J. Pan, J. T. Kwok, Q. Yang, et al. Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682, 2008.

[51] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760, 2010.

[52] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[53] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[54] Z. Peng, W. Zhang, N. Han, X. Fang, P. Kang, and L. Teng. Active transfer learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(4):1022–1036, 2019.

[55] A. Ramachandran, S. Gupta, S. Rana, and S. Venkatesh. Information-theoretic transfer learning framework for bayesian optimisation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 827–842. Springer, 2018.

[56] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao. When and how to transfer knowledge in dynamic multi-objective optimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2034–2041. IEEE, 2019.

[57] S. Ruder and B. Plank. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*, 2017.

[58] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[59] Y. Shi and F. Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. *arXiv preprint arXiv:1206.6438*, 2012.

[60] S. K. Smit and A. E. Eiben. Using entropy for parameter analysis of evolutionary algorithms. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 287–310. Springer, 2010.

[61] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[62] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye. A two-stage weighting framework for multi-source domain adaptation. *Advances in neural information processing systems*, 24:505–513, 2011.

[63] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2019.

[64] J. Talukdar, A. Biswas, and S. Gupta. Data augmentation on synthetic images for transfer learning using deep cnns. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 215–219. IEEE, 2018.

[65] B. Tan, Y. Zhang, S. J. Pan, and Q. Yang. Distant domain transfer learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[66] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[67] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3081–3088. IEEE, 2010.

[68] K. Torkkola and W. M. Campbell. Mutual information in learning feature transformations. In *ICML*, pages 1015–1022, 2000.

[69] L. Torrey and J. Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[70] M. Vargas, G. Fuertes, M. Alfaro, G. Gatica, S. Gutierrez, and M. Peralta. The effect of entropy on the performance of modified genetic algorithm using earthquake and wind time series. *Complexity*, 2018, 2018.

[71] K. Wang, J. Liu, and J.-Y. Wang. Learning domain-independent deep representations by mutual information minimization. *Computational intelligence and neuroscience*, 2019, 2019.

[72] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1406–1413, New York City, New York, USA, 1999. IEEE.

[73] Y. Wei, Y. Zheng, and Q. Yang. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1905–1914, 2016.

[74] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[75] Z. Wu, H. Jiang, K. Zhao, and X. Li. An adaptive deep transfer learning method for bearing fault diagnosis. *Measurement*, 151:107227, 2020.

[76] R. Xia, C. Zong, X. Hu, and E. Cambria. Feature ensemble plus sample selection: Domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18, 2013.

[77] D. Xu, Y. Luo, J. Luo, M. Pu, Y. Zhang, Y. Ha, and X. Luo. Efficient design of a dielectric metasurface with transfer learning and genetic algorithm. *Optical Materials Express*, 11(7):1852–1862, 2021.

[78] R. Yasarla, V. A. Sindagi, and V. M. Patel. Syn2real transfer learning for image deraining using gaussian processes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2726–2736, 2020.

[79] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114, 2004.

[80] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang. Heterogeneous transfer learning for image classification. In *AAAI*, volume 11, pages 1304–1309, 2011.