

Improving the Performance of Genetic Algorithms for Combinatorial Optimization Using Machine Learning for Knowledge Transfer

George Mweshi^a and Nelishia Pillay^b

Department of Computer Science, University of Pretoria, Pretoria, South Africa

Keywords: Genetic Algorithms, Combinatorial Optimization, Machine Learning, Classifiers.

Abstract: This study investigates improving the performance of genetic algorithms applied to the solution space using machine learning and knowledge transfer. Genetic algorithms are powerful techniques that have been successfully used to explore various problem spaces, such as solution space, program space, and heuristic space. Recently, researchers have found that transferring knowledge between these spaces can significantly enhance the quality of solutions and reduce computational costs. While this transfer of knowledge works well in program and heuristic spaces due to their indirect nature, it is more challenging in the solution space. This is because each problem in the solution space has its own unique representation, making it difficult to transfer knowledge effectively. This study explores how machine learning, specifically using classifiers, can help bridge this gap and facilitate knowledge transfer between different solution spaces. We train two classifiers, namely, Support Vector Machines and Random Forests, using data consisting of fitness landscape measures from a source genetic algorithm to determine if a chromosome is a local optimum or not. This information is then used during the execution of a target genetic algorithm to identify and remove potential local optima from the population. We tested this approach on two challenging optimization problems: the examination timetabling problem (ETP) and the capacitated vehicle routing problem (CVRP). Our results show that this method provides statistically significant improvements over genetic algorithms that do not use knowledge transfer, both in terms of solution quality and computational efficiency. Moreover, we found that random forests were more effective than support vector machines for transferring knowledge between the source and target genetic algorithms.


1 INTRODUCTION


Transfer learning involves transferring knowledge from a source optimization algorithm to a target optimization algorithm with the aim of improvements in the target domain (Zhuang et al., 2020). The benefits include improved quality of solutions in the source domain, reduced computational cost and a reduction in data needed to solve the problem in the target domain. This study focuses on transfer learning in evolutionary algorithms, in particular genetic algorithms

While transfer learning has been explored in genetic algorithms searching the program (Russell and Pillay, 2023), heuristic (Scheepers and Pillay, 2021), and design (Nyathi and Pillay, 2021) spaces, its application to genetic algorithms exploring the solution space for single-objective combinatorial optimization problems remains largely unexplored. The primary

challenge lies in determining an appropriate mapping from the source to the target domain, as chromosome representations in the solution space are problem-specific and not problem-domain independent, unlike in program, heuristic, and design spaces.

This study presents a new domain-independent approach for addressing the challenge of transferring knowledge between genetic algorithms using machine learning. Specifically, two classifiers, namely, Support Vector Machines (SVM) and Random Forests (RF) are trained to identify local optima based on fitness landscape measures such as ruggedness, neutrality, evolvability, and searchability collected from the source genetic algorithm. Once trained, these classifiers are applied to the target genetic algorithm to identify and remove potential local optima from its population. The approach was tested on two complex problems: the examination timetabling problem (ETP) and the capacitated vehicle routing problem (CVRP). The results indicate

^a  <https://orcid.org/0000-0002-3504-3700>

^b  <https://orcid.org/0000-0003-3902-5582>

that this knowledge transfer method improves performance compared to standard genetic algorithms, with Random Forests outperforming Support Vector Machines in both problem domains.

The key contribution of this study is:

- A machine learning approach for transfer learning in genetic algorithms exploring the solution space.

Although we focused on combinatorial optimization in this paper, the approach can also be applied to continuous optimization and machine learning problems. This research can also be seen as contributing to the growing effort by the computational intelligence community to use machine learning to improve the performance of optimization techniques.

2 BACKGROUND AND RELATED WORK

This section provides brief overview of genetic algorithms, transfer learning, fitness landscape analysis and machine learning techniques for classification. A discussion on some of the works that have investigated transfer learning in evolutionary algorithms is also provided.

2.1 Genetic Algorithms

Genetic algorithms (GAs) are optimization methods inspired by the principles of natural selection (Goldberg, 1989). They work by evolving a population of candidate solutions (which are represented as chromosomes) by iteratively applying the processes of selection, crossover (recombination), and mutation. These processes allow for the candidate solutions to be progressively refined until an optimal or satisfactory result is achieved. For a detailed overview of GAs, see (Goldberg, 1989). In this study, we used the generational GA as outlined in Algorithm 1.

2.2 Transfer Learning

Transfer learning (TL) is a technique that involves transferring knowledge gained from one domain or task (the source) to improve performance in a different but related domain or task (the target) (Zhuang et al., 2020). This technique is particularly useful when dealing with limited data in the target domain, as it leverages the knowledge acquired from the source domain to enhance the learning process.

In optimization, transfer learning can help by transferring learned features, patterns, or models from one optimization problem to another, with the goal

Algorithm 1: Pseudocode for a genetic algorithm.

Data: Population size N
Result: Individual with the highest fitness as the best solution
 Generate a population of individuals of size N ;
 Calculate the fitness of each individual in the population;
while *termination criterion is not met* **do**
 Select one or two individuals with the best fitness using a selection method;
 Generate new individuals for the next generation by applying genetic operators to previously selected individuals;
 Evaluate each new individual to determine its fitness;
 Replace all the individuals in the old population with the new individuals;
end
return *Individual with the highest fitness*

of improving the convergence times and quality of the solutions obtained by the optimization algorithms. TL has been successfully and widely applied in fields such as image recognition, natural language processing, and more recently, in various optimization algorithms. For an in-depth discussion of TL and its diverse applications, please refer to (Zhuang et al., 2020).

2.3 Fitness Landscape Analysis

Fitness landscape analysis (FLA) is an important technique for understanding how optimization algorithms, such as GAs, solve complex combinatorial optimization problems (Zou et al., 2022). The concept, introduced by Sewall Wright in 1932 (Wright et al., 1932) and formalized by Stadler (Stadler, 2002), can be described as follows:

$$(X, N, f) \quad (1)$$

Where, X represents the set of potential solutions from the decision variable space, N is the neighborhood operator that defines the relationships between solutions, and f is a function that maps solutions to their fitness values.

In general, FLA helps visualize and understand how different solutions relate to their fitness values. Each point in the fitness landscape corresponds to a potential solution, with its height reflecting the solution's quality. By analyzing this landscape, one can identify key features such as local and global optima, basins of attraction, and the overall structure of

the solution space. It also provides insights into the landscape’s ruggedness, neutrality, searchability, and evolvability (Zou et al., 2022). Some examples of fitness landscapes are shown in Figure 1.

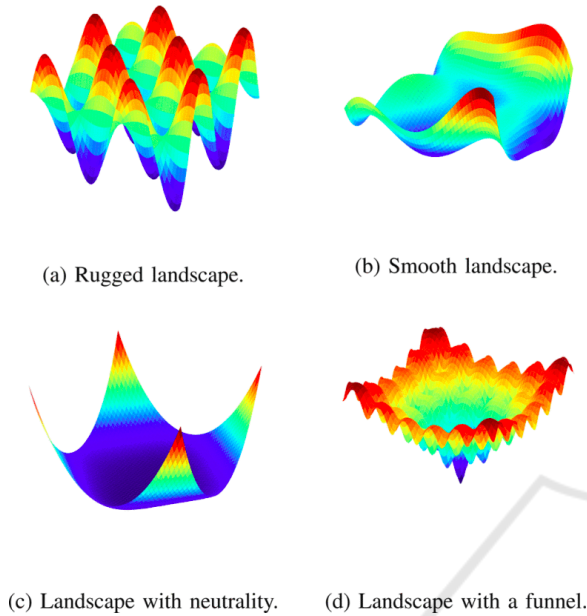


Figure 1: Some examples of fitness landscapes (Hassan and Pillay, 2022).

2.4 Machine Learning Techniques for Classification

Classification is a fundamental task in machine learning where the goal is to predict the categorical label of new observations based on training data. Among the various techniques used for classification, Support Vector Machines (SVMs) (Cervantes et al., 2020), (Sen et al., 2020) and Random Forests (RFs) (Schonlau and Zou, 2020) are two widely adopted methods due to their effectiveness and versatility.

2.4.1 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a powerful classification technique originally developed by Vladimir Vapnik and his colleagues (Cortes, 1995). The core idea of SVMs is to find a hyperplane that best separates different classes in the feature space. For linearly separable data, this involves identifying a hyperplane that maximizes the margin—the distance between the hyperplane and the nearest data points from each class, known as support vectors. Mathematically, this can be formulated as a quadratic optimization problem:

$$\text{minimize } \frac{1}{2} \|w\|^2 \quad (2)$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1 \quad \text{for all } i \quad (3)$$

where w represents the weights, x_i are the feature vectors, y_i are the class labels, and b is the bias term.

For non-linearly separable data, SVMs use kernel functions to transform the input space into a higher-dimensional space where a linear separation is possible. Common kernel functions include the polynomial kernel and the radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad (4)$$

where σ is a parameter that controls the spread of the kernel function.

SVMs are well-regarded for their generalization capabilities and have been successfully applied in various domains such as image recognition, text classification, and bioinformatics (Cortes, 1995).

2.4.2 Random Forests (RFs)

Random Forests (RFs) are an ensemble learning method that combines multiple decision trees to improve classification performance (Breiman and Cutler, 2001). A RF constructs multiple decision trees during training and outputs the class that is the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees. Each tree is built from a bootstrap sample of the training data, and at each split in the tree, a random subset of features is considered, which introduces additional randomness and improves the model’s generalization ability.

The algorithm can be summarized as follows:

1. Draw B bootstrap samples from the training data.
2. For each bootstrap sample, grow a decision tree using a random subset of features at each node.
3. Aggregate the predictions of all trees to make the final classification decision.

The ensemble approach of RFs helps to reduce variance and prevent overfitting, making them highly effective for a wide range of classification tasks, including those involving large and complex datasets (Breiman and Cutler, 2001).

2.5 Transfer Learning in Genetic Algorithms

As mentioned earlier, this paper investigates the application of transfer learning in genetic algorithms exploring the solution space for single-objective combinatorial optimization. Specifically, it employs machine learning, using classification to transfer knowledge between source and target genetic algorithms.

Previous research on transfer learning in evolutionary algorithms has predominantly focused on multi-objective optimization.

Jiang et al. (Jiang et al., 2020b) introduced MMTL-DMOEA, a memory-driven manifold transfer learning-based evolutionary algorithm for dynamic multi-objective optimization. By integrating memory mechanisms with manifold transfer learning, their algorithm significantly enhanced solution quality and reduced computational costs. Jiang et al. (Jiang et al., 2020a) proposed KT-DMOEA, a knee point-based imbalanced transfer learning method that transfers predicted knee points to reduce computational cost, leading to substantial improvements in solution quality.

Liu and Wang (Liu and Wang, 2021) combined a population prediction strategy (PPS) with a transfer learning-based dynamic multi-objective evolutionary algorithm (Tr-DMOEA) to address dynamic multi-objective optimization problems (DMOPs). This hybrid approach outperformed both PPS and Tr-DMOEA by effectively utilizing historical information to initialize populations in new environments.

Jiang et al. (Jiang et al., 2017) presented a framework integrating transfer learning with evolutionary algorithms to tackle DMOPs. By generating an initial population pool from past experiences and applying population-based evolutionary algorithms, this approach notably enhanced the performance of several well-known algorithms on benchmark functions.

Huang et al. (Huang et al., 2023) introduced a transfer learning-based evolutionary algorithm (TLEA) framework for multi-objective optimization problems. This framework decomposes complex problems into manageable subtasks, optimizing them collaboratively through transfer learning, and demonstrated superior performance on benchmark problems.

Zhang et al. (Zhang et al., 2023) transferred knowledge from an evolutionary algorithm-neural network hybrid solving low-order problems to the hybrid solving high-order functions, utilizing the optima found by the evolutionary algorithm and neural network model for low-order problems.

Unlike previous works, this study transfers knowledge from a genetic algorithm exploring the solution space for combinatorial optimization. A classifier is employed to learn and identify local optima in the source genetic algorithm, which are then removed from the population in the target genetic algorithm.

3 PROPOSED GA APPROACH FOR KNOWLEDGE TRANSFER

This section discusses the proposed GA approach used to solve the ETP and CVRP. We start by discussing the two problem domains and this is then followed by a discussion of how the GA was implemented. Finally, we explain how knowledge was transferred between the GAs to improve performance.

3.1 Problem Domains

This study considered two combinatorial optimization problems, namely, the examination timetabling problem (ETP) and the capacitated vehicle routing problem (CVRP).

3.1.1 Examination Timetabling Problem (ETP)

The ETP is a well-known optimization problem that involves scheduling exams within specified periods and rooms while strictly adhering to hard constraints and minimizing violations of soft constraints. In this study, we used the ITC 2007 examination timetabling benchmark set. The hard constraints for the benchmark set include:

- Ensuring that no student is scheduled to take more than one exam at the same time.
- Ensuring that the number of students assigned to a venue does not exceed its capacity.
- Guaranteeing that the duration of an exam fits within the allocated period.
- Respecting period-related constraints, such as scheduling one exam before another in the sequence.
- Satisfying room-related constraints, such as assigning exams to specific venues.

The soft constraints for the benchmark set include:

- Minimizing instances where students have to take two exams back-to-back or on the same day.
- Reducing the clustering of exams to ensure a more even distribution for students.
- Avoiding the scheduling of exams with mixed durations in the same period.
- Preferentially scheduling larger exams later in the timetable and minimizing the use of specific periods and rooms.

A feasible timetable is one that satisfies all hard constraints. The objective value of a timetable is calculated as the total cost of the violated soft constraints, as shown in Equation (5).

$$O_{ETP} = \sum_{i=1}^{n_{\text{soft}}} C_{\text{soft}}(i) \cdot S(i) \quad (5)$$

where, n_{soft} represents the total number of soft constraints, and $S(i)$ represents the number of violations for soft constraint i .

The goal is to create a timetable that violates no hard constraints while minimizing the number of soft constraint violations. The characteristics of the data instances in the ITC 2007 benchmark set are shown in Table 1.

Table 1: Characteristics of the ITC2007 ETP benchmark instances.

Instance	Exams	Students	Periods	Conflict Density	Rooms
1	607	7891	54	0.05	7
2	870	12743	40	0.01	49
3	934	16439	36	0.03	48
4	273	5045	21	0.15	1
5	1018	9253	42	0.009	3
6	242	7909	16	0.06	8
7	1096	14676	80	0.02	15
8	598	7718	80	0.05	8
9	169	655	25	0.08	3
10	214	1577	32	0.05	48
11	934	16439	26	0.03	40
12	78	1653	12	0.18	50

Conflict Density: number of conflicts / (number of exams)

3.1.2 Capacitated Vehicle Routing Problem (CVRP)

The CVRP on the other hand involves finding the most cost-effective set of routes for delivering goods to a group of customers while meeting strict constraints. In the study, we used the Christofides and Golden benchmark sets. The hard constraints associated with these benchmarks include:

- The vehicle must start its route at the depot and return to the depot after completing all deliveries.
- The total demand on a route must not exceed the vehicle's capacity.
- Each customer must be visited exactly once on a route.
- The duration of any route must not exceed a specified global maximum.

The main objective is to minimize the total cost of the route set. The objective value of a solution is calculated by summing the costs of all routes, which include the distances between customers and the service time for each customer, as shown in Equation (6).

$$O_{CVRP} = \sum_{i=1}^n \sum_{j=1}^n d_{ij} + \sum_{i=1}^n t_i \quad (6)$$

where n is the total number of customers.

The characteristics of the data instances in the Golden and Christofides benchmark sets are provided in Table 2 and Table 3 respectively.

Table 2: Characteristics of the Golden Benchmark set.

Instances	Capacity	Customers	Max. length	Service time	Vehicles
1	550	240	650	0	10
2	700	320	900	0	10
3	900	400	1200	0	10
4	1000	480	1600	0	12
5	900	200	1800	0	5
6	900	280	1500	0	8
7	900	360	1300	0	9
8	900	440	1200	0	11
9	1000	255	∞	0	14
10	1000	323	∞	0	16
11	1000	399	∞	0	18
12	1000	482	∞	0	19
13	1000	252	∞	0	27
14	1000	320	∞	0	30
15	1000	396	∞	0	34
16	1000	480	∞	0	38
17	200	240	∞	0	22
18	200	300	∞	0	22
19	200	360	∞	0	33
20	200	420	∞	0	41

Table 3: Characteristics of the Christofides Benchmark set.

Instances	Capacity	Customers	Max. length	Service time	Vehicles
1	160	51	∞	0	5
2	140	76	∞	0	10
3	200	101	∞	0	8
4	200	151	∞	0	12
5	200	200	∞	0	17
6	160	51	200	10	6
7	140	76	160	10	11
8	200	101	230	10	9
9	200	151	200	10	14
10	200	200	200	10	18
11	200	121	∞	0	7
12	200	101	∞	0	10
13	200	121	720	50	11
14	200	101	1040	90	11

3.2 Genetic Algorithm for ETP and CVRP

The GA used in this study is a generational algorithm where offspring replace parents in each generation (Goldberg, 1989). The pseudocode for the GA is shown in Algorithm 1.

3.2.1 Chromosome Representation

For the ETP, a chromosome typically represents an exam schedule, specifying the time slots and rooms assigned to each exam. In this study, the chromosome was encoded as an integer sequence with the value for each gene determined using a simple encoding function. The length of the chromosome was equal to the number of exams to be scheduled and these exams were arranged in ascending order within the chromosome. For example, the following chromosome '10 219 374 362 226 221' represents an exam schedule with 6 exams. The first exam is assigned an integer

value of 10, the second exam a value of 219, the third exam a value of 374 and so on. The integer values are obtained by using a simple encoding function shown in Equation 7 below:

$$\text{Int Value} = r\text{Index} \times \text{numPeriods} + \text{tsIndex} \quad (7)$$

where:

- $r\text{Index}$ refers to the index of the most suitable room for the exam,
- numPeriods represents the total number of available periods, and
- tsIndex indicates the most appropriate period for the exam.

The decoding process is as follows: each integer value in the chromosome is decoded into its corresponding room and period using Equation 8 and 9. The room index ($r\text{Index}$) is calculated by dividing the integer by the total number of periods, and the time slot index (tsIndex) is derived from the remainder of this division. This decoding allows us to accurately extract the room and period for each exam, thus reconstructing the exam schedule from the chromosome.

$$r\text{Index} = \left\lfloor \frac{\text{Int Value}}{\text{numPeriods}} \right\rfloor \quad (8)$$

$$\text{tsIndex} = \text{Int Value} \bmod \text{numPeriods} \quad (9)$$

For the Capacitated Vehicle Routing Problem (CVRP), the chromosome consisted of a vector of integers of length N , representing the number of customers to be served. Each gene in the vector corresponded to a given customer. The sequence of genes in the vector determined the service order of customers, and the set of customers that made up each route was limited by the capacity of the vehicles. That is, each customer was assigned to a specific vehicle and when the vehicle’s capacity was exceeded, a new route was started. The chromosome structure used is shown in Figure 2.

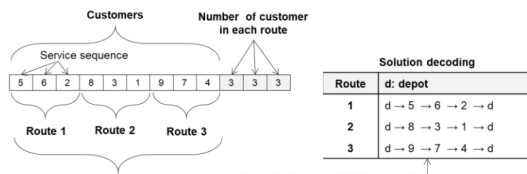


Figure 2: Solution encoding scheme using a vector of integers.

3.2.2 Population Initialization

We used two different approaches to generate the initial population for the ETP and CVRP. For the ETP, the initial population is generated randomly.

However, we implemented a special method called hard constraint solver, whose sole purpose was to generate random initial solutions that did not violate any hard constraints for the problem instances under consideration. This approach helped to significantly reduce the runtime for the GA. The pseudocode for the hard constraint solver is shown in Algorithm 2.

Algorithm 2: Hard Constraint Solver.

Data: List of exams, available rooms, available periods
Result: Solution satisfying hard constraints
 Begin with an empty scheduling solution;
 Randomize the order of exams;
while there are unscheduled exams **do**
 Choose an unscheduled exam;
 Verify its scheduling feasibility without violating constraints, prioritizing based on coincidence and precedence constraints;
 Identify suitable periods and rooms for scheduling the exam;
 Allocate the exam to a period and room;
 Update the solution with the scheduled exam;
end
return The updated scheduling solution

The classical and widely used Clarke-Wright (Lysgaard, 1997) heuristic method was used to generate the initial population for the CVRP.

3.2.3 Fitness Evaluation

We used Equation (5) and Equation (6) as fitness functions for the ETP and CVRP respectively.

3.2.4 Selection Method

Tournament selection (Yadav and Sohal, 2017) was used to select parents for the genetic operators.

3.2.5 Genetic Operators

We used the two-point crossover operator for the ETP. This approach was favored for its simplicity in implementation and its ability to introduce increased diversity among the offspring compared to the one-point crossover. Additionally, we used the partially-mapped crossover (PMX) and order crossover (OX) operators (Ahmed et al., 2023) for the CVRP. For mutation, we used the following operators: Swap, Two-opt, scramble, inversion and displacement operators. More information on these mutation operators can be found in (Daglayan and Karakaya, 2016).

3.2.6 Replacement Strategy

Although elitism is a commonly used strategy in GAs, it was not used in this study in order to maintain focus on the effects of machine learning-based knowledge transfer. Future work will explore the incorporation of elitism to examine its impact on preserving the best individuals and whether it enhances the GA's ability to learn local optima more effectively.

3.2.7 Termination Criteria

For both problems, we set the termination criteria to be the maximum number of generations. We experimented with various values for this and the best values are shown in Table 4.

3.3 Knowledge Transfer in the GA Solution Space

This section describes how knowledge about the local optima in the solution space was transferred between the source GA and the target GA.

3.3.1 Fitness Landscape Measures

The following fitness landscape measures: autocorrelation, correlation length, neutrality, and evolvability were used to classify whether a particular element of the population was a local optimum or not. These measures were selected because they provide single numerical values for each solution in the population, thereby making it easier to quantify each solution in terms of its fitness landscape measures.

- **Autocorrelation and Correlation Length** (Brandt, 2001), (Merkuryeva and Bolshakovs, 2011): These measures were used to assess the ruggedness of the fitness landscape. Ruggedness refers to the variability or "roughness" of the fitness landscape. High ruggedness implies the presence of many local optima, making it challenging for optimization algorithms to find the global optimum. Generally, autocorrelation examines the similarity between values at different points, while correlation length measures the distance over which points are correlated. A rapidly decaying autocorrelation function suggests high ruggedness and a high likelihood of local optima. Conversely, a short correlation length implies a rugged landscape with frequent local optima, indicating that a solution in such a region is likely near a local optimum.
- **Neutrality**: Neutrality refers to the extent to which small changes in a solution do not result in

changes in fitness. In a neutral landscape, many neighboring solutions have the same or similar fitness. The Average Neutrality Ratio (ANR) (Vanneschi et al., 2006) was used to quantify neutrality. The neutrality ratio of a point is the proportion of neutral neighbors to the total neighbors of that point, and ANR is the average of these neutrality ratios across the landscape. Higher ANR values indicate a more neutral landscape.

- **Evolvability**: Evolvability reflects the ability of a population to improve its fitness over generations. A highly evolvable landscape allows the algorithm to navigate towards higher fitness regions efficiently. Evolvability was measured using the Accumulated Escape Probability (AEP) (Lu et al., 2011), derived from the Fitness Probability Change (FPC), which is the mean of the average escape probabilities from different points. Higher AEP values indicate a more evolvable landscape.

Local optima are identified by comparing a solution's fitness with the fitness of its neighboring solutions within the fitness landscape. A solution is deemed to be a local optimum if its fitness is lower (in our case) than that of all its neighbors. This approach helps the GA avoid being trapped in suboptimal regions of the search space. While this method may resemble hill-climbing, our focus was on detecting and eliminating such local optima through the learning process, rather than performing traditional hill-climbing.

3.3.2 Source Genetic Algorithm

During execution of the source GA, a dataset was created. The dataset was built by collecting solutions found in the final population at different stages of the algorithm's run. We specifically looked at solutions found after 10, 15, 20, 25, 30, 35, 40, and 45 generations. Each data instance in this dataset represented a single solution. It consisted of two parts as shown in Figure 3:

- **Features**: A set of independent variable values derived from fitness landscape measures. These features described the characteristics of the solution.
- **Target**: A corresponding target value indicating whether the solution is a local optimum (yes or no). We used the following binary values (1 for local optimum, 0 for not).

3.3.3 Model Training and Testing

The machine learning algorithms (SVM and RF) were trained using the data obtained after executing the

1	Fitness	Autocorrelation	Correlation Length	ANR	AEP	Local Optimum
2	0.92	0.85	5	0.1	0.3	1
3	0.87	0.75	10	0.05	0.4	0
4	0.9	0.8	7	0.08	0.35	1
5	0.83	0.7	8	0.07	0.45	0
6	0.95	0.88	4	0.12	0.25	1
7	0.85	0.78	9	0.09	0.38	0

Figure 3: Example of a dataset with FLA features.

source Genetic Algorithm (GA) (refer to 3.3.2). The dataset was then split into training and testing sets with a 80-20 ratio. The training set was used to train the models, while the testing set was used to evaluate their performance. Min-Max Normalization was applied to ensure that all input variables contributed equally to the model training. Grid search (Sun et al., 2021) was used to find the optimal hyperparameters both SVM and RF. The best hyperparameter values for the two classifiers are listed in Table 4.

Model Evaluation. The performance of the models was assessed using the following metrics:

- **Accuracy.** The proportion of correctly predicted instances out of the total instances.
- **Precision.** The ratio of true positive predictions to the total predicted positives.
- **Recall (Sensitivity or True Positive Rate).** The ratio of true positive predictions to the total actual positives. It shows the model's ability to find all the positive instances.
- **F1 Score.** The harmonic mean of precision and recall, used to balance the trade-off between the two when they are not equally important

The next section describes how the models produced were used in the target genetic algorithm.

3.3.4 Target Genetic Algorithm

Knowledge from the source GA was transferred to the target GA in the form of two classifier models with each model consisting of FLA10, FLA15, FLA20, FLA25, FLA30, FLA35, FLA40 and FLA45 sub models. If a solution was determined to be a local optimum, it was subsequently eliminated from the population. This was done in order to allow the target GA to look for better solutions by avoiding those solutions that were likely to lead to a local optima. In addition, by eliminating these solutions from the population, our hope was that the convergence time for the target GA would be improved as well.

4 EXPERIMENTAL SETUP

This section provides an overview of the experimental setup used to assess the effectiveness of the GA-FLA.

4.1 Source and Target Problem Instances

The problem instances in Section 3.1 were divided into source and target sets. The GA was executed for each problem instance in the source set, generating data over 10, 15, 20, 25, 30, 35, 40 and 45 generations for training the classification models.

For the ETP, we opted for a source domain set comprising instances that were computationally less challenging to solve:

- Source set instances: 4, 8, 11, 12
- Target set instances: 1, 2, 3, 5, 6, 7, 9, 10

For the CVRP, we implemented a simple K-Means clustering method (Sinaga and Yang, 2020) on the benchmark datasets to group instances sharing common characteristics. This strategy was preferred over random selection, as our initial trials showed that random selection frequently resulted in poor results. Subsequently, we selected representative instances from each cluster to bolster the source domain.

- Golden
 - Source set instances: 1, 4, 8, 12, 18, 20
 - Target set instances: 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 17, 19
- Christofides
 - Source set instances: 3, 4, 9, 10, 13
 - Target set instances: 1, 2, 5, 6, 7, 11, 14

4.2 Experiments

In order to evaluate the performance of GA-FLA, the following two experiments were conducted:

- Experiment 1: GA-FLA Comparison with Data from Different Generations - Compares the performance of the GA-FLA with the classifier learning with data from 10, 15, 20, 25, 30, 35, 40, and 45 generations. We also include an ensemble model which intelligently combines all the sub models (i.e. 10, 15, 20, 25, 30, 35, 40, and 45 models) using majority voting.
- Experiment 2: GA and GA-FLA Performance Comparison - This experiment compares the performance of the genetic algorithm without transfer learning (GA) to the that of the genetic algorithm with transfer learning (GA-FLA) for both the problem domains.

Due to the stochastic nature of the algorithms 30 runs, each with a different random number generator seed is performed for each problem instance for both the source and target GAs.

It is important to note that fairness in computational budget was ensured by using the same number of generations (50) for both the standard GA and the GA-FLA. While additional machine learning-based steps were introduced for GA-FLA, the overall evaluation time included this training and classification process. Therefore, the reported times reflect the actual computational cost of both approaches fairly.

4.3 Statistical Tests

We conducted hypothesis testing using the Z statistic to assess whether there was a significant difference in the performance between GA-FLA and a basic GA. A confidence level of 95% (i.e., p-values below 0.05 are statistically significant) was used for both statistical tests.

4.4 GA and Classifier Parameters

The GA approach was implemented using the ECJ (Evolutionary Computation in Java) toolkit (Luke, 1998) while the Weka (Bouckaert et al., 2016) toolkit was employed for the SVM and RF implementations.

To identify the most effective parameter values, we adopted an empirical approach involving systematic experimentation and iterative adjustment of parameters. The best parameter values obtained during the fine-tuning process are listed in Table 4.

Table 4: Best parameter values.

Technique	Parameter	Value
Parameter	Value	
GA	Population Size	100
	Crossover Rate	0.8
	Mutation Rate	0.1
	Selection Operator	Tournament
	Tournament Size	3
	Crossover Operator	Two-Point / OX
	Mutation Operator	Two-Opt
SVM	Kernel Type	RBF (Radial Basis Function)
	Cost	1.0
	Gamma	0.01
	Cache Size	100
RF	Number of Trees	100
	Maximum Depth	0 (unlimited)
	Minimum Size for Split	2

4.5 Technical Specifications

The the source and target GAs were executed on a computing system with the following hardware

configuration: An Intel Core i7 octa-core processor (clocked at 2.8 GHz), 16GB RAM, NVidia GeforceRTX 2080 GPU and a 500GB SSD.

5 RESULTS AND DISCUSSION

This section compares the performance of the GA-FLA on the two experiments outlined in section 4.2.

5.1 Experiment 1: GA-FLA Comparison with Data from Different Generations

This section compares the performance of GA-FLA using classifiers trained with data from different stages of the source GA. While data was collected at multiple intervals, specifically at generations 10, 15, 20, 25, 30, 35, 40, and 45, only the results from the best-performing generations are shown in tables 5 to 10 for clarity and relevance.

Table 5: GA-FLA Comparison with RF for ETP.

Instance	Gen30			Ensemble		
	Best	Average	Time(secs)	Best	Average	Time(secs)
1	4928	5401	2300	4792	5120	2300
2	482	512	2300	432	490	2300
3	7930	8359	2300	7831	8210	2300
5	2647	2901	3600	2602	2890	3600
6	25926	26870	2300	25730	26640	2300
7	4060	4410	3600	3922	4200	3600
9	972	1080	2300	965	1010	2300
10	13390	14214	3600	13222	14080	3600

Table 6: GA-FLA Comparison with SVM for ETP.

Instance	Gen30			Ensemble		
	Best	Average	Time(secs)	Best	Average	Time(secs)
1	5170	5860	2300	4930	5310	2300
2	572	599	2300	490	540	2300
3	8190	8730	2300	8002	8480	2300
5	2710	3050	3600	2678	2950	3600
6	26065	26941	2300	25901	26872	2300
7	4100	4560	3600	4042	4431	3600
9	985	1092	2300	974	1040	2300
10	13572	14650	3600	13410	14360	3600

The best objective value and average objective value over the 30 runs is listed. It is evident from the tables that the ensemble classifier outperforms the individual classifiers. This was found to be statistically significant at a 95% level of confidence for the ETP. However, for the CVRP, the results were found to be significant only at the 90% level of confidence. There is also no difference in computational cost despite the improvement in performance. These results also indicate that the RF classifier produced better results than the SVM classifier.

While the maximum number of generations was set to 50, which may appear conservative for large

Table 7: GA-FLA Comparison with RF for CVRP Golden Data Set.

Instance	Gen30			Ensemble		
	Best	Average	Time(secs)	Best	Average	Time(secs)
2	8591.5	8810.6	372	8560.1	8790.3	372
3	11508.6	12090.3	406	11445.7	11847.5	406
5	7909.5	8330.7	367	7742.5	8120.4	367
6	8872.9	9005.3	405	8765.8	8944.0	405
7	11064.2	11428.1	509	10922.3	11260.2	509
9	710.7	795.2	468	685.4	750.1	468
10	799.8	880.6	490	741.6	830.5	490
11	1103.4	1198.1	578	1024.9	1130.3	578
13	1005.2	1150.7	703	939.1	970.2	703
14	1240.2	1301.4	512	1190.8	1242.8	512
15	1529.8	1608.4	950	1465.7	1550.1	950
16	1689.2	1790.5	1081	1670.2	1710.8	1081
17	1065.3	1102.1	641	920.5	973.2	641
19	1540.5	1608.7	394	1490.2	1580.3	394

Table 8: GA-FLA Comparison with SVM for CVRP Golden Data Set.

Instance	Gen30			Ensemble		
	Best	Average	Time (secs)	Best	Average	Time (secs)
2	8760.3	8970.1	372	8630.2	8891.2	372
3	12029.5	12292.7	406	12009.4	12200.9	406
5	8220.6	8430.2	367	8040.1	8310.9	367
6	9001.4	9371.2	405	8937.7	9230.4	405
7	11640.1	11822.5	509	11368.6	11629.8	509
9	790.5	810.4	468	710.6	770.3	468
11	1105.4	1203.6	578	1073.5	1150.7	578
13	1022.1	1085.4	703	957.3	990.6	703
14	1303.8	1392.7	512	1243.3	1347	512
15	1594.3	1640.2	950	1530.6	1597.1	950
16	1684.6	1765.2	1081	1680.3	1740.8	1081
17	1104.6	1197.5	641	933.0	1003.2	641
19	1646.2	1699.5	394	1500.1	1621.6	394

Table 9: GA-FLA Comparison with RF for CVRP Christofides Data Set.

Instance	Gen30			Ensemble		
	Best	Average	Time (secs)	Best	Average	Time (secs)
1	651.7	694.1	100	630.7	660.3	100
2	850.3	870.2	270	847.5	864.1	270
5	1420.5	1543.2	430	1399.2	1520.8	430
6	846.8	870.4	220	790.7	843.1	220
7	965.2	970.3	286	960.4	970.1	286
8	920.6	1080.4	301	911.3	1042.1	301
11	1280.8	1450.3	343	1224.4	1408.4	343
12	970.4	1022.1	323	949.2	1000.5	323
14	960.5	990.2	318	928.9	967.1	318

Table 10: GA-FLA Comparison with SVM for CVRP Christofides Data Set.

Instance	Gen30			Ensemble		
	Best	Average	Time (secs)	Best	Average	Time (secs)
1	680.2	690.3	100	650.4	667.3	100
2	855.4	890.6	270	852.1	870.5	270
5	1550.1	1599.5	430	1432.4	1499.7	430
6	840.1	877.3	220	822.2	869.4	220
7	970.4	995.7	286	968.3	985.4	286
8	930.1	1010.2	301	915.7	999.6	301
11	1170.5	1255.8	343	1120.6	1245.2	343
12	910.9	1090.1	323	899.4	995.3	323
14	972.4	1008.3	318	962.8	980.5	318

problem instances, this choice was made to assess the GA’s ability to find better solutions within a limited computational budget. The results indicate that the GA-FLA effectively reached better optima within this constraint. Future work will explore running the

algorithms with higher generation limits to evaluate whether additional improvements can be achieved.

5.2 Experiment 2: GA and GA-FLA Performance Comparison

This section compares the performance of the GA, i.e. the GA without transfer learning to the GA-FLA with the best classifier, namely, the ensemble, from the previous section. Tables 11, 12 and 13 compare the best results obtained by both GA and GA-FLA for the ETP, CVRP Golden and CVRP Christofides datasets respectively. The percentage of improvement ($\Delta(\%)$), if any, is calculated using Eq. (10).

$$\Delta(\%) = \frac{best_{GA} - best_{GA-FLA}}{best_{GA}} * 100 \quad (10)$$

Where $best_{GA}$ is the objective value of the best solution obtained by the GA approach and $best_{GA-FLA}$ is the objective value of the best solution obtained by the GA-FLA approach

The results indicate that GA-FLA outperforms GA without an increase in computational cost for both problems. This result was found to be significant at the 95% level of significance

Table 11: GA vs. GA-FLA with RF for ETP.

Instance	GA			GA-FLA			$\Delta(\%)$
	Best	Average	Time(secs)	Best	Average	Time(secs)	
1	6770	7430	2600	4792	5120	2300	29.2
2	793	990	2600	432	490	2300	45.5
3	8769	9320	2600	7831	8210	2300	10.7
5	3413	3802	4000	2602	2890	3600	23.8
6	28330	30450	2600	25730	26640	2300	9.2
7	5535	7020	4000	3822	4200	3600	30.9
9	1092	1580	2600	965	1010	2300	11.6
10	14053	16860	4000	13222	14080	3600	5.9

Table 12: GA vs. GA-FLA with RF for CVRP Golden Data Set.

Instance	GA			GA-FLA			$\Delta(\%)$
	Best	Average	Time(secs)	Best	Average	Time(secs)	
2	9560.9	9822.8	975	8560.1	8790.3	372	10.5
3	12251.5	12508.6	992	11445.7	11847.5	406	6.6
5	9005.7	9339.2	696	7742.5	8120.4	367	14
6	9937.7	10154.9	861	8765.8	9154.9	405	11.8
7	12368.6	13629.8	987	10922.3	11860.2	509	11.7
9	1009.0	1115.3	883	685.4	750.1	468	32
10	1001.5	1020.8	889	741.6	830.5	490	26
11	1373.5	1459.5	1017	1024.9	1130.3	578	25.4
13	1342.7	1440.5	1197	939.1	970.2	703	30.1
14	1597.2	1626.5	924	1190.8	1242.8	512	25.4
15	1881.1	1927.4	1130	1465.7	1550.1	950	22.1
16	1761.5	1809.7	1362	1670.2	1710.8	1081	5.2
17	1533.0	1596.8	881	920.5	973.2	641	40
19	1852.2	1921.1	532	1490.2	1580.3	394	19.5

In order to understand the performance improvement achieved through transfer learning, we analyzed the progression of fitness values for a problem instance from some benchmark sets. Figure 4 and Figure 5 shows how the fitness values of the two GAs evolved over generations for the elected instances from the datasets.

Table 13: GA vs. GA-FLA with RF for CVRP Christofides Data Set.

Instance	GA			GA-FLA			Δ(%)
	Best	Average	Time(secs)	Best	Average	Time(secs)	
1	788.1	861.4	129	630.7	660.3	100	20
2	963.7	993.7	288	847.5	864.1	270	12
5	1997.3	2061.4	496	1399.2	1520.8	430	30
6	896.1	1008.7	247	790.7	843.1	220	11.8
7	1097.9	1165.8	304	960.4	970.1	286	12.5
8	1366.4	1475.8	325	911.3	1042.1	301	33.3
11	2124.8	2263.1	367	1224.4	1408.4	343	42.4
12	1204.7	1289.4	352	949.2	1000.5	323	21.2
14	1252.2	1321.1	332	928.9	967.1	318	25.8

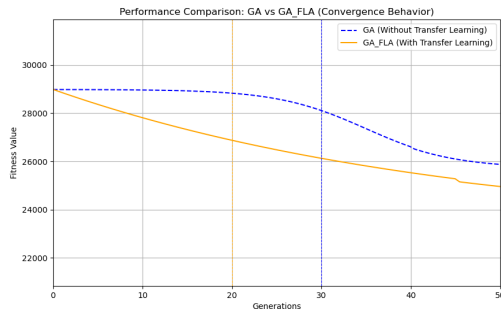


Figure 4: Comparison of fitness progression of the GAs for ETP ITC2007 dataset instance 6.

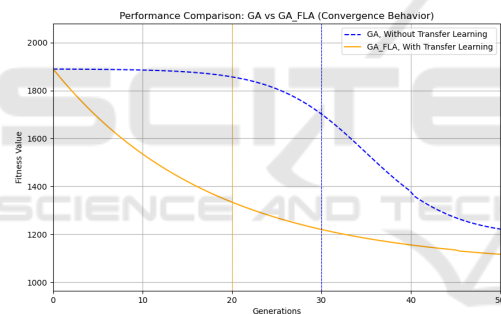


Figure 5: Comparison of fitness progression of the GAs for Golden dataset instance 14.

From the two figures it is evident that the GA-FLA reaches a better optimum quicker than the GA, i.e., the use of the knowledge transfer approach has enabled the GA to move to an area of the search space with better solutions more quickly.

6 CONCLUSION

The main aim of this study was to investigate the transfer of knowledge in GAs exploring the solution space. While the concept of knowledge transfer has been effectively employed in GAs exploring the heuristic, program and design spaces, this has not been investigated for genetic algorithms exploring the solution space for single objective discrete op-

timization. The reason for this is the challenge of the solution space consisting of different representations for different problem instances. In this study we turned to machine learning to overcome this challenge. A classifier was trained on data consisting of fitness landscape measures from the source GA. This trained classifier was then used in the during the execution of the target GA to eliminate solutions leading to local optima from the population. The proposed approach was evaluated for discrete optimisation on a benchmark set for the ETP and two benchmark sets for the CVRP. For all problem instances (GA-FLA) was found to outperform the genetic algorithm without knowledge transfer (GA) with a reduction in computational cost. The reason for this performance was that the GA-FLA was also able to move to an area with better optima quicker than the GA as it avoided areas likely to lead to poor results. The study also revealed that using an ensemble of classifiers, trained on data from different subsets of generations in the source GA, was the most effective. Furthermore, RF were found to perform better than SVM. Overall, the results showed that the incorporation of knowledge transfer mechanisms in a GA results in improvements not only in the quality of solutions obtained, but also the convergence time.

Future work will explore two main areas: (1) the incorporation of elitism into the genetic algorithm to investigate whether preserving the best individuals enhances the ability to find global optima, and (2) the use of a higher number of generations to assess if further improvements can be achieved in both solution quality and convergence time.

ACKNOWLEDGMENTS

This work was funded as part of the MultiChoice Research Chair in Machine Learning at the University of Pretoria, South Africa.

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Number 138150). Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

REFERENCES

Ahmed, Z. H., Al-Otaibi, N., Al-Tameem, A., and Saudagar, A. K. J. (2023). Genetic crossover operators for the capacitated vehicle routing problem. *Computers, Materials & Continua*, 75(1).

- Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., and Scuse, D. (2016). Weka manual for version 3-9-1. *University of Waikato: Hamilton, New Zealand*, pages 1–341.
- Brandt, H. (2001). Correlation analysis of fitness landscapes.
- Breiman, L. and Cutler, R. (2001). Random forests machine learning [j]. *journal of clinical microbiology*, 2:199–228.
- Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215.
- Cortes, C. (1995). Support-vector networks. *Machine Learning*.
- Daglayan, H. and Karakaya, M. (2016). The impact of crossover and mutation operators on a ga solution for the capacitated vehicle routing problem. *Universal Journal of Engineering Science*, 4(3):39–44.
- Goldberg, D. (1989). Genetic algorithms in search, optimization and machine learning. reading, ma: Addison-wesley professional.
- Hassan, A. and Pillay, N. (2022). Automated design of hybrid metaheuristics: A fitness landscape analysis. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.
- Huang, J., Wen, J., Chen, L., and Liu, H.-L. (2023). Transfer learning based evolutionary algorithm framework for multi-objective optimization problems. *Applied Intelligence*, pages 1–20.
- Jiang, M., Huang, Z., Qiu, L., Huang, W., and Yen, G. G. (2017). Transfer learning-based dynamic multiobjective optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 22(4):501–514.
- Jiang, M., Wang, Z., Hong, H., and Yen, G. G. (2020a). Knee point-based imbalanced transfer learning for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):117–129.
- Jiang, M., Wang, Z., Qiu, L., Guo, S., Gao, X., and Tan, K. C. (2020b). A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning. *IEEE Transactions on Cybernetics*, 51(7):3417–3428.
- Liu, Z. and Wang, H. (2021). 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.
- Lu, G., Li, J., and Yao, X. (2011). Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In *Evolutionary Computation in Combinatorial Optimization: 11th European Conference, EvoCOP 2011, Torino, Italy, April 27-29, 2011. Proceedings 11*, pages 108–117. Springer.
- Luke, S. (1998). ECJ evolutionary computation library. Available for free at <http://cs.gmu.edu/~eclab/projects/ecj/>.
- Lysgaard, J. (1997). Clarke & wright’s savings algorithm. *Department of Management Science and Logistics, The Aarhus School of Business*, 44.
- Merkuryeva, G. and Bolshakovs, V. (2011). Benchmark fitness landscape analysis. *International Journal of Simulation Systems, Science and Technology*, 12(2):38–45.
- Nyathi, T. and Pillay, N. (2021). On the transfer learning of genetic programming classification algorithms. In Aranha, C., Martín-Vide, C., and Vega-Rodríguez, M. A., editors, *Theory and Practice of Natural Computing*, pages 47–58, Cham. Springer International Publishing.
- Russell, J. and Pillay, N. (2023). A selection hyper-heuristic for transfer learning in genetic programming. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation, GECCO ’23 Companion*, pages 631–634, New York, NY, USA. Association for Computing Machinery.
- Scheepers, D. and Pillay, N. (2021). A study of transfer learning in a generation constructive hyper-heuristic for one dimensional bin packing. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.
- Schonlau, M. and Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, 20(1):3–29.
- Sen, P. C., Hajra, M., and Ghosh, M. (2020). Supervised classification algorithms in machine learning: A survey and review. In *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pages 99–111. Springer.
- Sinaga, K. P. and Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727.
- Stadler, P. F. (2002). Fitness landscapes. In *Biological evolution and statistical physics*, pages 183–204. Springer.
- Sun, Y., Ding, S., Zhang, Z., and Jia, W. (2021). An improved grid search algorithm to optimize svr for prediction. *Soft Computing*, 25:5633–5644.
- Vanneschi, L., Pirola, Y., and Collard, P. (2006). A quantitative study of neutrality in gp boolean landscapes. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 895–902.
- Wright, S. et al. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution.
- Yadav, S. L. and Sohal, A. (2017). Comparative study of different selection techniques in genetic algorithm. *International Journal of Engineering, Science and Mathematics*, 6(3):174–180.
- Zhang, T.-T., Hao, G.-S., Lim, M.-H., Gu, F., and Wang, X. (2023). A deep hybrid transfer learning-based evolutionary algorithm and its application in the optimization of high-order problems. *Soft Computing*, pages 1–12.
- Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.
- Zou, F., Chen, D., Liu, H., Cao, S., Ji, X., and Zhang, Y. (2022). A survey of fitness landscape analysis for optimization. *Neurocomputing*, 503:129–139.