

Transfer Learning in Genetic Algorithms for Combinatorial Optimization Using Fitness Lineage

George Mweshi^[0000–0002–3504–3700], Nelishia Pillay^[0000–0003–3902–5582], and
Thambo Nyathi^[0000–0003–4676–6063]

Department of Computer Science University of Pretoria, Pretoria, South Africa
`nelishia.pillay@up.ac.za` `t.nyathi@cs.up.ac.za`

Abstract. Transfer learning improves the efficiency and performance of evolutionary algorithms, delivering higher-quality solutions while reducing computational time and resources. Transfer learning in genetic algorithms works by transferring knowledge from a source genetic algorithm trained on solving simpler problems, to a target genetic algorithm for solving more complex problems. Transfer learning in genetic algorithms is challenging as transferring knowledge, i.e., chromosomes representing solutions, is not effective as the target genetic algorithms will be solving a different problem instance. This study examines transferring problem-independent knowledge, namely, knowledge regarding the search. Machine learning, namely, predictors, are used to transfer knowledge regarding whether an element of the population will lead to a local optimum or not. The paper introduces the concept of fitness lineage to identify local optima. The proposed approach is evaluated for combinatorial optimisation using two problem domains, namely, the examination timetabling problem (ETP) and the capacitated vehicle routing problem (CVRP). For both problems, knowledge is collected from the source genetic algorithms solving simpler problem instances to the target genetic algorithms solving more complex problems. The knowledge transferred is a predictor to identify local optima. The genetic algorithms with transfer learning (GATL) outperformed the genetic algorithms without transfer learning (GA) for all problem instances for both problems. Furthermore, the proposed approach outperformed previous work using machine learning to transfer knowledge in genetic algorithms.

Keywords: Knowledge Transfer, Genetic Algorithms, Fitness Lineage, Predictors, Examination Timetabling, Capacitated Vehicle Routing.

1 Introduction

Transfer learning in evolutionary algorithms uses knowledge gained from a source evolutionary algorithm, that solves simple computationally cheap problems, to a target genetic algorithm that solves more complex computationally expensive problems[8]. The reasoning is that the transfer of knowledge will result in the

target evolutionary algorithms finding better solutions at a lower cost. This research has focused predominantly on the program, heuristic and design spaces [7, 10, 11] as knowledge transfer is feasible due to the representation used in these spaces being problem independent, e.g., program, heuristic or design. However, in the case of knowledge transfer in evolutionary algorithms exploring the solution space, such transfer of knowledge is not feasible as the knowledge is problem instance specific. For example, if a genetic algorithm is used to evolve solution timetables for one problem instance, any domain-specific knowledge learned will not be relevant to solving a different problem instance. Hence, this study investigates transferring domain-independent knowledge, namely, information about the search, instead of solution knowledge in genetic algorithms exploring the solution space.

Research in this area has been limited, with only one other study examining domain-independent knowledge transfer in genetic algorithms for single objective optimisation [6]. In this study the domain-independent knowledge transferred is a random forest classifier to identify potential local optima in the target genetic algorithm. Fitness landscape features are used as attributes by the classifier to identify potential local optima. Integrating transfer learning enabled the genetic algorithm to outperform the standard genetic algorithm, achieving superior solution quality at a lower computational cost. The research presented in this paper provides an alternative to identifying local optima. It introduces the concept of *fitness lineage* to identify local optima. Furthermore, predictors are used to predict fitness based on the fitness lineage.

The source genetic algorithms (SGAs) are used to solve simpler problems that are computationally cheaper. While solving these problems, a training set is created based on fitness lineage to train a predictor to identify potential local optima. The predictor is then used in the target genetic algorithm (TGA) to eliminate potential local optima. The baseline performance of the standard genetic algorithm (GA) is compared to that of the genetic algorithm using transfer learning (GATL) in solving two combinatorial optimisation problems, namely, the examination timetabling problem (ETP) along with the capacitated vehicle routing problem (CVRP). GATL was found to outperform the GA on all problem instances for both problems. GATL also outperformed the classifier approach in previous work used for domain-independent transfer knowledge in genetic algorithms for combinatorial optimisation in terms of both solution quality and computational cost. The research contributes the following:

- Transfer learning in genetic algorithms using fitness lineage and predictors to improve solution quality using domain-independent knowledge for combinatorial optimisation.
- Transfer learning in genetic algorithms using fitness lineage and predictors to reduce computational cost using domain-independent knowledge for combinatorial optimisation.

The problem domains used to evaluate GA and GATL are presented in section 2. The genetic algorithms implemented for solving the ETP and CVRP are

described in section 3. The transfer learning approach using predictors (GATL) is presented in section 4. The experimental setup used to evaluate the performance of GA and GATL in solving the ETP and CVRP is discussed in section 5. The performance of both approaches is compared in section 6. Section 7 highlights the findings of the study and future research directions.

2 Problem Domains

This section outlines the two problem domains used to evaluate GATL: the examination timetabling problem and the capacitated vehicle routing problem. This section describes both the problem domains that GATL is evaluated on, namely, the examination timetabling problem and the capacitated vehicle routing problem. These problem domains have been chosen to allow for comparison with previous work employing machine learning for knowledge transfer in genetic algorithms for combinatorial optimisation problems. Furthermore, genetic algorithms are not the best performing technique for these problem domains, hence there is room for improvement.

2.1 Examination Timetabling Problem (ETP)

The ETP requires assigning exams in periods and locations without violating the hard constraints and minimising soft constraint violations [9]. Various benchmark sets have been utilised for this problem; the most recent and most constrained dataset is the ITC 2007 exam timetabling benchmark dataset, which was created based on real-world institutions for the second international timetabling competition [5]. The hard constraints governing this benchmark include:

- No student is assigned to take multiple exams simultaneously.
- Students assigned to a venue should not exceed the venue capacity.
- The duration of an exam must not exceed that of the period it has been scheduled in.
- Period-specific constraints, such as, scheduling one exam before another in sequence.
- Room-specific constraints, e.g., assigning examinations to specific venues.

The soft constraints defined for the benchmark comprise of:

- Minimising two examinations in a row on the same day for students.
- The even distribution for students.
- Minimising the occurrence of exams with differing durations held within the same period.
- Scheduling larger exams later in the timetable
- Minimising the use of specific periods
- Minimising the use of specific rooms.

A valid timetable fulfils all hard constraints. The primary goal is to generate an examination timetable that strictly adheres to all hard constraints while minimising soft constraint violations. Details of the ITC 2007 benchmark set are presented in Table 1.

Table 1. Attributes of the ITC2007 ETP benchmark problems sets

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

$$\text{Conflict Density} = (\text{number of conflicts}) / (\text{number of exams})$$

2.2 Capacitated Vehicle Routing Problem (CVRP)

The CVRP involves finding the most cost-effective set of routes for delivering goods to a group of customers while meeting the problem constraints [13]. In the study, we used the Christofides[2] and Golden[3] benchmark sets. The hard constraints associated with these benchmarks are:

- Each vehicle must depart from the central depot and return there once all deliveries are finished.
- The total demand on any single route cannot 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 minimise the total cost of the set of routes. Details regarding the data instances for the Christofides and Golden benchmark sets are provided in table 2 and table 3, respectively.

3 Genetic Algorithm(GA)

This section presents the genetic algorithm approach for both problem domains. The generational genetic algorithm illustrated in Algorithm 1 is applied to both problem domains. It begins by generating a random initial population, which undergoes iterative refinement across a specified number of generations. Parents are chosen for genetic operations using tournament selection [14]. This involves randomly selecting s elements from the population, where s is a parameter, and returning the best element as a parent. Section 3.1 and section 3.2 provide the specific details of the genetic algorithm for the ETP and the CVRP.

Table 2. 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

Table 3. 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

Algorithm 1 Generational Genetic Algorithm

```

0: Create the initial population randomly
0: while Termination criterion is not met do
0:   Evaluate the population
0:   Select parents
0:   Create offspring
0: end while
0: return The updated solution. =0

```

3.1 Genetic Algorithm for the ETP

An element of the population is a timetable encoded as a string of integers to reduce computational cost. Each integer represents the position of the examination in the timetable in terms of the period and room it is allocated to. Each timetable in the initial population is created to ensure feasibility which is maintained throughout the evolutionary process. Hence, the fitness is determined by the total count of soft constraints violated. The fitness of a chromosome is calculated according to equation 1:

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

where $C_{\text{soft}}(i)$ is the penalty associated with the violation of soft constraint i and $S(i)$ is the number of violations for that constraint. In this formulation, n_{soft} represents the total number of soft constraints.

A chromosome is considered valid only if it satisfies all hard constraints, i.e., it incurs no penalty from hard constraint violations. The goal is to evolve an exam schedule that minimises the total penalty from soft constraint violations while ensuring that all hard constraints are strictly adhered to.

Offspring of the next generation were created using crossover and mutation. One of the following mutation operators was randomly selected for mutation:

- Swap mutation - Two randomly selected examinations exchange their rooms or periods.
- Three-opt mutation - The periods or rooms of three pairs of examinations are swapped.
- Scramble mutation- A number of examinations n are randomly selected and assigned to new periods or rooms.
- Inversion mutation - A sequence of examinations in a period are randomly selected and the order of the examinations reversed.

Two-point crossover operator is used. This operator basically involves selecting two crossover points in the parent chromosomes. The genes situated between these two points in one parent are exchanged with the genes between the same points in the other parent, yielding two offspring.

3.2 Genetic Algorithm for the CVRP

A chromosome is a route, where a gene corresponds to a customer. Customers are visited in the order in which they appear in the chromosome. Each customer can only be visited once, and there is a limit on the number of vehicles. The Clarke-Wright algorithm [4] is used to create each chromosome.

The fitness measure of a chromosome is the total distance travelled by vehicles in delivering goods to customers and not exceeding vehicle capacity using the formula in equation 2:

$$F_{\text{CVRP}} = \sum_{i=1}^n \sum_{j=1}^n d_{ij} + \sum_{i=1}^n t_i \quad (2)$$

where n denotes the total customer count, d_{ij} indicates the distance separating customers i and j , and t_i specifies the service time required for customer i . As with the ETP the genetic operators used are to produce offspring are mutation and crossover. The following mutation operators are used:

- Swap mutation - Two genes in a chromosome are randomly selected and swapped.
- Two-opt mutation - Two pairs of connected customers in a route are randomly selected and swapped.
- Scramble mutation - A sequence of customers in a route are randomly selected and the order of the customers shuffled.
- Inversion Mutation - In inversion mutation, a segment of the route is chosen, and the order of customers within that segment is then reversed.

Two crossover operators from [1] are used:

- Partially-mapped crossover(PMX) - which selects two points from each parent, and the sequence of customers between both points are swapped.
- Ordered crossover (OMX) - also selects two crossover points in each of the parents, and the segment demarcated by the points and the first gene remains fixed. The remaining genes are swapped.

4 Genetic Algorithm with Transfer Learning(GATL)

This section presents GATL. The transfer learning process involves three steps:

- Step 1: Data collection from the source genetic algorithm(SGA)
- Step 2: Predictor training
- Step 3: Predictor application in the target genetic algorithm(TGA)

The following sections describe each of these steps.

4.1 Data Collection from the SGA

We introduce the concept of *fitness lineage* to train predictors to transfer knowledge. The fitness lineage of every individual in the starting population is tracked across all generations. The fitness lineage of an individual I in the initial population is the fitness of its successive offspring from the first generation, after the initial population, to the i th generation of a run, where i is less than the maximum number of generations. In the case of crossover, where two offspring are produced, the fitness of the fitter offspring is included in the lineage. The lineage is used in Step 2 to train the predictor. The predictor is trained to predict the next fitness value, i.e., the fitness of the next offspring, in the fitness lineage.

4.2 Predictor Training

The fitness lineage data that is obtained from using the source genetic algorithm to solve simpler problems is used to train a predictor to predict fitness given a series of previous fitness values, i.e. the previous lineage. Max-min normalisation was used to normalise the data for the different problems solved. The data was split into 80% for training and testing was 20% . Three predictors were evaluated for this purpose, namely, linear regression, support vector regression and random forest regression. Initial trials found that the random forest predictor was the most effective, so it was used in the study. This is consistent with the study in [6]. The data was collected for different numbers of generations, namely, 10, 15, 20, 25, 30, 35, 40 and 45, where the maximum number of generations for a run is 50 generations. Different predictors were created for the different numbers of predictors providing a total of 8 predictors P10, P15, P20, P25, P30, P35, P40 and P45. These predictors were included in an ensemble predictor to predict fitness. The output of the ensemble is the average of the output of each predictor. Trial runs revealed that ensemble was more effective than the separate predictors. The ensemble predictor is used in the target genetic algorithms to eliminate local optima.

4.3 Target Genetic Algorithm

The generational genetic algorithm depicted in Algorithm 1 is the target genetic algorithm with elements of the population at set number of generations corresponding to the training of the generations that the predictor was trained on. The ensemble predictor was used to predict the fitness for the different number of generations, 10 to 45. If the maximum number of generations had not been achieved, the corresponding predictor was not evoked. If the predicted fitness of an individual did not improve for a predefined number of predictions g which is set as a parameter, the individual is marked as a local optimum and removed from the population. Elements removed were replaced using crossover and mutation applied to the best performing elements of the population. It is randomly determined whether to perform mutation or crossover and tournament selection for the parent/s is restricted to the top $k\%$ of elements of the population, where k is a parameter value.

5 Experimental Setup

This section details the experimental designed to assess the performance of GATL.

5.1 Source and Target problem data instances

The problem instances presented in Section 2 were divided into source and target sets. For the ETP, the source domain set consisted of the 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 K-Means clustering method, [12] was applied to the benchmark datasets to group instances sharing common characteristics. This strategy was preferred over random selection, as initial trials showed that random selection frequently resulted in poor results. Representative instances from each cluster were selected.

- 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

5.2 Statistical Tests

Thirty one runs, with different random number generator seeds, were performed for both GA-PFM and the GA. Hypothesis testing, using the Z statistic, was used to evaluate the statistical significance of the results comparing the performance of GA-PFM and the GA without transfer learning. A 95% level of confidence was used. The Z-statistic was used as the study compares two approaches and to ensure that the distribution is normal 31 runs were performed.

5.3 GA and RF Parameter Tuning

Grid search was used for parameter tuning. The values found for the random forest regressor and genetic algorithms are depicted in Table table 4 and table 5, respectively.

Table 4. Random Forest Parameter

Parameter	Value
Number of Trees	100
Batch size	100
Maximum Depth	0 (unlimited)
Number of Iterations	100

5.4 Technical specifications

Both GATL and GA algorithms were implemented in Python from scratch. For the random forest and grid search, the scikit-learn library was used. The models were run on a computing system with an Intel® Core™ i7-9750H $\times$ 12 processor operating at 2.6 GHz with 16 GB of DDR4 RAM. The system was also equipped with a 500 GB NVMe Solid State Drive (SSD) as well as an NVIDIA GeForce GTX 1660 graphics processing unit (GPU).

Table 5. GA parameter values

Parameter	Value
Population Size	100
Crossover Rate	0.8
Mutation Rate	0.2
Selection Operator	Tournament Selection
Tournament size	3
Crossover Operator	Two-Point Crossover/OX
Mutation Operator	Two-Opt Mutation
Max. Generations	50

Table 6. GATL Parameters

Parameter	Value
k (Elite percentage of the population)	5%
g (Number of generations to identify local optimum)	10

6 Results and Discussion

This section compares the performance of the genetic algorithms without transfer learning, GA, to the genetic algorithm with transfer learning, GATL, for the ETP and CVRP problems. The objective values and runtimes in seconds are listed in tables 10, 11 and 12. For all problem instances, GATL was found to outperform the GA. These results were found to be statistically significant at the 95% level of significance. Furthermore, the computational cost for the GA is greater than the GATL for all problem instances.

Table 7. Comparison of GA and GATL for ETP ITC2007 Instances Performance

Instance	GA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
1	6770	7430	2600	4690	4910	1800
2	793	990	2600	408	451	1800
3	8769	9320	2600	7710	8064	1800
5	3413	3802	4000	2480	2630	2800
6	28 330	30 450	2600	25550	26 050	1800
7	5535	7020	4000	3860	4023	2800
9	1092	1580	2600	948	980	1800
10	14 053	16 860	4000	13083	13 083	2800

We also compare the performance of GATL with the current state-of-the-art approach using machine learning for transfer learning in a genetic algorithm in [6], namely, GA-FLA. As can be seen from table 10, 11 and 11, GATL produces better results than GA-FLA. The runtimes are generally lower, with just two

Table 8. Comparison of GA and GATL for Golden Instances

Instance	GA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
2	9560.9	9822.8	975	8410.3	8720.5	372
3	12 251.5	12 508.6	992	10997.8	11 320.4	406
5	9005.7	9339.2	696	7525.6	7850.1	367
6	9937.7	10 154.9	861	8550.7	8720.6	405
7	12 368.6	13 629.8	987	10710.8	11 020.3	509
9	1009.0	1115.3	883	595.3	635.2	468
10	1001.5	1020.8	889	748.2	785.4	490
11	1373.5	1459.5	1017	935.8	965.1	578
13	1342.7	1440.5	1197	895.7	920.3	703
14	1597.2	1626.5	924	1125.6	1160.4	512
15	1881.1	1927.4	1130	1395.2	1430.6	950
16	1761.5	1809.7	1362	1645.7	1675.8	1081
17	1533.0	1596.8	881	720.6	740.4	641
19	1852.2	1921.1	532	1375.6	1418.2	394

Table 9. Comparison of GA and GATL for Christofides Instances

Instance	GA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
1	788.1	861.4	129	524.6	560.2	100
2	963.7	993.7	288	836.2	870.1	270
5	1997.3	2061.4	496	1295.7	1340.5	220
6	896.1	1008.7	247	565.8	595.3	286
7	1097.9	1165.8	304	913.4	950.1	301
8	1366.4	1475.8	325	865.9	900.2	343
11	2124.8	2263.1	367	1045.6	1080.2	323
12	1204.7	1289.4	352	820.3	850.2	318
14	1252.2	1321.1	332	870.4	900.3	318

problem instances with slightly higher runtimes than GA-FLA, which are not statistically significant.

Table 10. Comparison of GA-FLA and GATL for ETP ITC2007 Instances Performance

Instance	GA-FLA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
1	4792	5120	2300	4690	4910	1800
2	432	490	2300	408	451	1800
3	7831	8210	2300	7710	8064	1800
5	2602	2890	3600	2480	2630	2800
6	25 730	26 640	2300	25550	26 050	1800
7	3822	4200	3600	3860	4023	2800
9	965	1010	2300	948	980	1800
10	13 222	14 080	3600	13083	13 083	2800

Table 11. Comparison of GA-FLA and GATL for Golden Instances

Instance	GA-FLA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
2	8560.1	8790.3	372	8410.3	8720.5	372
3	11 445.7	11 847.5	406	10997.8	11 320.4	406
5	7742.5	8120.4	367	7525.6	7850.1	367
6	8765.8	9154.9	405	8550.7	8720.6	405
7	10 922.3	11 860.2	509	10710.8	11 020.3	509
9	685.4	750.1	468	595.3	635.2	468
10	741.6	830.5	490	748.2	785.4	490
11	1024.9	1130.3	578	935.8	965.1	578
13	939.1	970.2	703	895.7	920.3	703
14	1190.8	71 242.8	512	1125.6	1160.4	512
15	465.7	1550.1	950	1395.2	1430.6	950
16	1670.2	1710.8	1081	1645.7	1675.8	1081
17	920.5	973.2	641	720.6	740.4	641
19	1490.2	1580.3	394	1375.6	1418.2	394

7 Conclusion and Future Work

This study has proposed a machine learning for transfer learning in genetic algorithms as an alternative to the current state-of-the-art. The study evaluates

Table 12. Comparison of GA and GATL for Christofides Instances

Instance	GA-FLA			GATL		
	Best	Average Time (secs)		Best	Average Time (secs)	
1	630.7	660.3	100	524.6	560.2	100
2	847.5	864.1	270	836.2	870.1	270
5	1399.2	1520.8	430	1295.7	1340.5	220
6	790.7	843.1	220	565.8	595.3	286
7	960.4	970.1	286	913.4	950.1	301
8	911.3	1024.1	301	865.9	900.2	343
11	1224.4	1408.4	343	1045.6	1080.2	323
12	949.2	1000.5	323	820.3	850.2	318
14	928.9	967.1	318	870.4	900.3	318

random forest predictors to identify potential local optima using fitness lineage. Data is gathered from the source genetic algorithms to train the predictor, and the predictor is used in the target genetic algorithm to eliminate potential local optima. The proposed approach was found to outperform the genetic algorithm without transfer learning for both the capacitated vehicle routing and examination timetabling problems. It was also found to outperform the current state-of-the-art machine learning approach for transfer learning in genetic algorithms for the same problem domains. Furthermore, the computational cost of the approach was less than both these approaches. Future work will explore the application of GATL to other optimisation problems and investigate the use of other machine learning approaches, e.g., unsupervised and reinforcement learning. It will also compare random removal, stagnation-based removal, and elite replacement to the proposed approach.

7.0.1 Acknowledgments. This research is supported, in part, by the National Research Foundation (NRF) of South Africa (Grant No. 138150). Any views, conclusions presented in this publication are strictly those of the author and do not necessarily reflect the official position of the NRF.

Code and Data Availability

The datasets used are publicly available datasets and references to these have been cited in the paper.

Bibliography

- [1] Ahmed, Z.H., Al-Otaibi, N., Al-Tameem, A., Saudagar, A.K.J.: Genetic crossover operators for the capacitated vehicle routing problem. *Computers, Materials & Continua* **75**(1) (2023)
- [2] Lobit, H.: Christofides dataset-capacitated vehicle routing problem. <http://www.vrp-rep.org/datasets/item/2014-0002.html> (2014)
- [3] Lobit, H.: Golden datasets- capacitated vehicle routing problem. <http://www.vrp-rep.org/datasets/item/2014-0004.html> (2014)
- [4] Lysgaard, J.: Clarke & wright's savings algorithm. *Department of Management Science and Logistics, The Aarhus School of Business* **44**, 1–7 (1997)
- [5] McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A.J., Gaspero, L.D., Qu, R., Burke, E.K.: Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS J. on Computing* **22**(1), 120–130 (Jan 2010). <https://doi.org/10.1287/ijoc.1090.0320>, <https://doi.org/10.1287/ijoc.1090.0320>
- [6] Mweshi, G., Pillay, N.: Improving the performance of genetic algorithms for combinatorial optimization using machine learning for knowledge transfer. pp. 363–374 (01 2024). <https://doi.org/10.5220/0013084200003837>
- [7] Nyathi, T., Pillay, N.: On the transfer learning of genetic programming classification algorithms. In: Aranha, C., Martín-Vide, C., Vega-Rodríguez, M.A. (eds.) *Theory and Practice of Natural Computing*. pp. 47–58. Springer International Publishing, Cham (2021)
- [8] Pillay, N.: Transfer learning in evolutionary spaces. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. p. 1005–1016. GECCO '24 Companion, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3638530.3648431>, <https://doi.org/10.1145/3638530.3648431>
- [9] Qu, R., Burke, E., Mccollum, B., Merlot, L., Lee, S.: A survey of search methodologies and automated system development for examination timetabling. *J. Scheduling* **12**, 55–89 (02 2009). <https://doi.org/10.1007/s10951-008-0077-5>
- [10] Russell, J., Pillay, N.: A selection hyper-heuristic for transfer learning in genetic programming. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. p. 631–634. GECCO '23 Companion, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3583133.3590686>, <https://doi.org/10.1145/3583133.3590686>
- [11] Scheepers, D., Pillay, N.: 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)*. pp. 1–7 (2021). <https://doi.org/10.1109/SSCI50451.2021.9660092>
- [12] Sinaga, K.P., Yang, M.S.: Unsupervised k-means clustering algorithm. *IEEE access* **8**, 80716–80727 (2020)

- [13] Toth, P., Vigo, D., Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications, Second Edition. Society for Industrial and Applied Mathematics, USA (2014)
- [14] Yadav, S.L., Sohal, A.: Comparative study of different selection techniques in genetic algorithm. International Journal of Engineering, Science and Mathematics **6**(3), 174–180 (2017)