A Meta-Genetic Algorithm for Hybridizing Metaheuristics

Ahmed Hassan(✉) and Nelishia Pillay

University of KwaZulu-Natal, Pietermaritzburg 3209, South Africa
ahmedhassan@aims.ac.za, pillayn32@ukzn.ac.za

Abstract. The research presented in this paper forms part of the initiative aimed at automating the design of intelligent techniques to make them more accessible to non-experts. This study focuses on automating the hybridization of metaheuristics and parameter tuning of the individual metaheuristics. It is an initial attempt at testing the feasibility to automate this design process. A genetic algorithm is used for this purpose. Each hybrid metaheuristic is a combination of metaheuristics and corresponding parameter values. The genetic algorithm explores the space of these combinations. The genetic algorithm is evaluated by applying it to solve the symmetric travelling salesman problem. The evolved hybrid metaheuristics are found to perform competitively with the manually designed hybrid approaches from previous studies and outperform the metaheuristics applied individually. The study has also revealed the potential reusability of the evolved hybrids. Based on the success of this initial study, different problem domains shall be used to verify the automation approach to the design of hybrid metaheuristics.

Keywords: Metaheuristics · Hybrid metaheuristics · Meta-genetic algorithm

1 Introduction

Hybrid metaheuristics have proven to be effective at solving combinatorial optimization problems [4,27]. However, designing hybrid metaheuristics is a very challenging task [4,27] involving various critical design decisions, including which metaheuristics to combine, how to combine these and which parameter values to use for each metaheuristic. The research presented in this paper aims at investigating the possibility of automating these design decisions.

A genetic algorithm is used to automate these design decisions. The metaheuristics are combined linearly and each chromosome is a sequence of metaheuristics, with each gene containing a character representing a metaheuristic and the relevant parameter values for that metaheuristic. A steady-state genetic algorithm employing tournament selection, crossover and mutation is used to explore the space of metaheuristics and their parameter values. The metaheuristics comprising the chromosomes include iterated local search, tabu search, simulated annealing and a memetic algorithm.

© Springer International Publishing AG 2017
E. Oliveira et al. (Eds.): EPIA 2017, LNAI 10423, pp. 369–381, 2017.
DOI: 10.1007/978-3-319-65340-2_31

The automatically designed hybrid metaheuristics (ADHMs) evolved by the meta-genetic algorithm (MGA) are evaluated using the symmetric traveling salesman problem (TSP) and their performance is compared to manually designed hybrid approaches. The ADHMs are found to be very competitive to the manually designed approaches. The ADHMs are also found to outperform the metaheuristics when they are used separately. The main contributions of this research are:

- This study has shown a simple, yet effective, approach to the automation of hybrid metaheuristics. This has been illustrated using the symmetric traveling salesman problem, however the approach can be applied to solving any combinatorial optimization problem; see Sect. 7.
- The research has shown the potential of automating the design of hybrid metaheuristics. The approach performs competitively with carefully crafted, manually designed hybrid heuristics. The approach is also found to have better performance than the standard metaheuristic when applied individually.

The rest of the paper is organized as follows. In Sect. 2, the motivation of the paper is stated. Section 4 describes the proposed meta-genetic algorithm. The experimental setup is specified in Sect. 5. In Sect. 6, the results are presented and discussed. Section 7 provides the conclusion and the future work.

2 Motivation

The best known results in many optimization problems are found by hybrid metaheuristics [27]. The design of hybrid metaheuristics is challenging, requiring expert knowledge in algorithm design, data structures and statistics [4]. The design of metaheuristics involves many hard-to-make design choices such as whether to use a single method or to hybridize a number of methods; which methods/components to include; how to synthesize the hybridization; how to prune the space of all possible design choices; how to configure the hybrid method and/or each of its components. The main aim of this research is to investigate the possibility of automating this design process which will give researchers the time to focus on other aspects, with a long-term goal of providing tools for non-experts to use, hence facilitating multidisciplinary research.

Analogous to the way experts come up with hybrid solvers, in this paper the design phase is separated from the application phase. The design phase involves designing a hybrid metaheuristic for solving a particular optimization problem. In the application phase, the algorithm crafted in the design phase is applied to the optimization problem at hand. The design phase is usually conducted manually by the researcher. In this study we investigate automating this process thereby relieving the researcher from carrying it out manually.

Please note that the aim of this research is not to compete with or improve on the state of the art techniques for solving a particular problem. The main focus is to show how the design process can be automated and to investigate the difference in performance of the automatically designed and manually designed

hybrid metaheuristics, with the aim of producing hybrids that perform at least as good as the manually designed hybrids.

3 Related Work

The automation of the design of metaheuristics is tackled from many aspects using different approaches: algorithm selection [25], algorithm portfolios [15], reactive search [2], automatic algorithm configuration [18] and hyper-heuristics [5].

In a narrower sense, prior work includes that of Adriaensen et al. [1] in which a focused ILS is used for automating hyper-heuristics. Kanda et al. [19] use meta-learning to choose, based on the problem features, the most effective metaheuristic. Bhanu and Gopalan [3] use a great deluge hyper-heuristic to control a set of hybrid genetic algorithms. Grobler et al. [16] use a selection hyper-heuristic to manage a number of population-based metaheuristics. Maashi et al. [21] use a choice function hyper-heuristic to intelligently select the most appropriate multi-objective evolutionary algorithm at each point of solving the problem. Pillay [24] uses an evolutionary algorithm hyper-heuristic to evolve sequences of classical artificial intelligent search methods.

To the best of our knowledge, this the first study which uses a multipoint metaheuristic to automatically hybridize and tune single point and population-based metaheuristics. The success of using genetic algorithms for parameter tuning in evolutionary algorithms [13] as well as determining the control flow in evolutionary algorithms [11] motivates the use of the genetic algorithm in this paper. Please note that the subject of this work is to hybridize metaheuristics; not to select the most suitable metaheuristic based on the instance features.

4 Meta-genetic Algorithm for Hybridizing Metaheuristics

The MGA is a steady state genetic algorithm evolving a population of metaheuristic hybrids consisting of the standard metaheuristics (SMHs); namely, simulated annealing (SA), tabu search (TS), iterated local search (ILS) and a memetic algorithm (MA). Along with finding appropriate metaheuristic hybrids, the MGA also finds suitable parameter settings for the SMHs in an online fashion.

4.1 Standard Metaheuristics

Please refers to [14] for the description of the SMHs used in this paper. The details that are specific to our implementation are only presented here.

ILS: The perturbation of the ILS is done by executing random moves for a number of times determined by the perturbation strength. The local search used by the ILS is the best improvement local search which makes the best move at each step and stops when there is no further improvement. The acceptance criterion accepts improving moves only.

TS: The TS used in this paper is the best improvement tabu search which makes the best non-tabu move at each iteration. A move is allowed by the aspiration criterion if it is the best move seen so far in the neighborhood and it produces a solution better than the best solution. The tabu list is ruled by: first enters, first leaves. The tabu condition is defined in Sect. 5.2.

SA: The temperature decreases geometrically, i.e. $T_{k+1} = c \times T_k$ where $c \in (0, 1)$ is the cooling rate. The length of the repetition schedule is equal to the size of the neighborhood. The probability function is defined as:

$$\Pr(s, s', T) = \exp\left(\frac{-100 \times [f(s') - f(s)]\,/f(s')}{T}\right),$$

where $f(.)$ denotes the objective function, T denotes the temperature and s is the current solution and s' is a candidate solution.

MA: The MA is a steady state genetic algorithm combined with local search. At each generation, the maximum preservative crossover operator [22] is used to produce one offspring. Then, the local search is applied to the offspring. The worse individual in the population is replaced by the offspring if the offspring is better. The local search is the first improvement local search which makes the first improving move it finds and stops when there is no further improvement.

4.2 Chromosome Representation and Initial Population Generation

The chromosomes are represented by strings of the form $\mathtt{h}_1 : \mathtt{s}_1, \mathtt{h}_2 : \mathtt{s}_2, \ldots, \mathtt{h}_n : \mathtt{s}_n$ where `n` is the chromosome length and each gene is of the form $\mathtt{h}_i : \mathtt{s}_i$ for `i = 1, 2, ..., n` where $\mathtt{h}_i$ can be simulated annealing `S`, tabu search `T`, iterated local search `I`, or the memetic algorithm `M` and $\mathtt{s}_i$ is the specification for running $\mathtt{h}_i$. For instance, if $\mathtt{h}_i$ is the tabu search `T`, then $\mathtt{s}_i$ could be `(10, 75)` which means running the tabu search for `10` iterations and the length of tabu list is `75`. An example of a chromosome is `M:(50,100,2),S:(45, 0.5, 0.95)` which is interpreted as running the memetic algorithm for `50` generations with a population of size of `100` and a tournament of size `2` followed by running simulated annealing with an initial temperature of `45`, a final temperature of `0.5` and a cooling rate of `0.95`.

The chromosomes of the initial population are created at random where each gene is created by selecting one of the SMHs along with its parameters at random. The parameters are chosen from predetermined ranges. The length of each chromosome is also randomly chosen. The shortest chromosome is of length one and the longest chromosome is bounded by a limit. Duplicates are not allowed in the initial population.

4.3 Fitness Evaluation and Selection

The fitness of each chromosome is calculated by executing the specified SMH at each gene using the corresponding specification. Each chromosome is evaluated on one problem instance. All elements of a population are applied to the same

initial solution of the same problem instance. The execution of the chromosomes is handled sequentially, i.e. once the execution of a gene ends, the execution of the subsequent gene starts using the output of the preceding gene as an input. The execution of the chromosome is terminated once an optimal solution is found or until the last gene is executed. The fitness of the chromosome is the cost of the best solution found by the chromosome. To facilitate the transition from one gene to the next, the best solution found by the preceding gene is passed to the subsequent gene if it encodes a single point search. When the subsequent gene encodes a population-based method, seventy percent of the initial population is made of randomly created solutions and the rest of the population is comprised of the solutions created by the preceding genes inserted with equal proportions.

Tournament selection is used which involves choosing a few individuals at random from the population which compete based on fitness. The fittest individual is the winner of the tournament. The winner of the tournament is used to create offspring.

4.4 Regeneration

The crossover operator is used to produce the offspring. Two crossover points are chosen independently at random in both parents and the chromosomal substrings are swapped at the crossover points so that two offspring are generated. Then the offspring are mutated at one gene chosen at random. The mutation is done by choosing a SMH along with its parameter values at random. The two fittest individuals out of the two parents and the two offspring survive and they are inserted into the next generation.

5 Experimental Setup

5.1 Evaluation of the MGA

The symmetric TSP is used to evaluate the performance of the proposed MGA. The TSP has been used extensively to benchmark newly proposed methods due to its rich complexity and wide applications.

The proposed MGA is an automated approach for designing metaheuristic hybrids and thus it is compared to recently proposed manually designed hybrid systems; namely, the method of Wu et al. [28] (GCGA), the method of Créput and Koukam [7] (MSOM), the method of Chen and Chien [6] (GSAACPSO) and the two methods of Lin et al. [20] (AHSATS-DCM and AHSATS-2OPT).

The problem set consists of 30 problem instances chosen from TSPLIB.[1] The number of cities in these instances ranges from 51 to 724. It is worth noting that although instances of these sizes can be solved up to optimality using exact techniques, they are still challenging for recent hybrid metaheuristics and approximate hybrid methods; see for instance [9,23,26]. Furthermore, the aim of the proposed study is not to develop a specialized large-scale TSP solver.[2]

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[2] Large-scale TSP solvers require special data structures and heuristics [17] which falls out of the scope of this study.

5.2 TSP Specific Details

The solutions are tours represented as permutations of integers where each city is identified by a unique number from 0 to $n-1$ assuming there are n cities. The initial solutions are created uniformly at random using the algorithm of Durstenfeld [12]. The length of tours is calculated as specified in the TSPLIB. The 2-opt operator [8] is used to facilitate the transition from one solution to a neighboring solution during the search. The perturbation of the ILS described in Sect. 4.1 is done by executing random 2-opt moves determined by the perturbation strength. The tabu condition of the TS described in Sect. 4.1 is based on the 2-opt moves. A 2-opt move is declared tabu if both cities involved in the 2-opt move are in the tabu list.

5.3 Parameter Settings

The parameter values of the SMHs are chosen by the MGA from specific ranges. The ranges are chosen to be wide enough to give the MGA the flexibility of determining the appropriate values for each parameter. The ranges of the parameters for each SMH are documented online.[3]

5.4 Experiments

There are three experiments conducted.

Experiment I: The main aim of Experiment I is to compare the automated design with the manual design. The automatically designed hybrid metaheuristics evolved by the MGA are compared to the manually designed approaches mentioned in Sect. 5.1 in terms of the best and the average performance. Please note that there is a clear separation between the design phase and the application phase; thus, unlike the hyper-heuristic studies [5], the MGA replaces the human designer and thus it is not directly compared with the published methods; rather, the performance of the hybrid metaheuristics that are designed by the MGA is compared with the performance of the manually designed hybrid methods. The Friedman test is used to detect whether there are significant differences in the performance of the ADHMs and the other methods. Then, the Wilcoxon signed ranks test is used to perform the pairwise comparisons involving the ADHMs and the manually designed approaches. The Holm procedure is used to control the family-wise error associated with multiple comparisons.

Experiment II: The aim of Experiment II is to compare the performance of the hybrid solvers to the individual metaheuristics, i.e. the TS, ILS, SA and MA. Each metaheuristic is tuned automatically offline using Iterated Race for Automatic Algorithm Configuration (irace).[4] The details of the offline parameter tuning can be found online.[5] The performance of the evolved hybrid metaheuristics is

[3] https://www.dropbox.com/s/2cyeywtxvd56rjr/parameter_settings.rtf?dl=0.
[4] http://iridia.ulb.ac.be/irace/.
[5] https://www.dropbox.com/s/01imrwvtc48ce18/offline_tuning.rtf?dl=0.

compared to each of the metaheuristics individually applied to solve the problem instance. The Friedman test and the Holm procedure are used.

Experiment III: This experiment is an initial investigation of the reusability of the ADHMs. A set of 22 instances is used. The instances are divided into "hidden" and "seen" instances. The hidden instances are grouped into two classes: ClassA comprised of 10 instances with sizes less than 300 cities and ClassB comprised of 10 instances with sizes larger than 300 cities. For each class, one "seen" instance is used by the MGA to evolve a hybrid solver. Then, that hybrid solver is used to solve all the hidden instances belonging to the corresponding class. The best and average objective value over 30 runs are used to evaluate the performance of the hybrid solvers on the hidden instances.

5.5 Implementation Platform

The simulations are run on a cluster consisting of Intel 5th generation CPUs (2.6 GHz). The cluster is interconnected with FDR 56 GHz InfiniBand. The OS is CentOS 7.0. The programming language is the OpenJDK Java version 1.7.0_111.

6 Results and Discussion

6.1 Experiment I

In Experiment I, the performance of the ADHMs is compared to that of the manually designed approaches. The best and the mean performance of the ADHMs over 30 independent runs are shown in Table 1. For each method shown in the table, the first column gives the percentage deviation of the best solutions from the best known solutions where the percentage deviation (denoted by Δ) of a solution S from the best known solution S^* is defined as

$$\Delta = 100 \times \frac{S - S^*}{S^*}. \tag{1}$$

The second column gives the mean performance. The third column gives the runtime measured in seconds. From Table 1, the ADHMs find the best known solutions for 22 instances out of 30. The ADHMs perform better in terms of both the best and mean performance compared to GCGA, MSOM, GSAACPSO; perform slightly better than AHSATS-2OPT in terms of the best performance and slightly worse in terms of the mean performance; perform equally with the AHSATS-DCM in terms of the best performance and slightly worse in terms of the mean performance.

The statistical analysis is used to detect whether the differences in the performance of the ADHMs and the other published methods are significant. The Friedman test is used. The average rankings computed by the Friedman test are AHSATS-CDM: 1.53, AHSATS-2OPT: 2.22, ADHMS: 2.25, GSAACPSO: 4.06, MSOM: 5.44 and GCGA: 5.5. High ranks correspond to poor performance. The Friedman

Table 1. Comparison of the performance of the ADHMS with previously published work. BKS stands for the best known solution.

Instance	BKS	MGA			AHSATS-DCM			AHSATS-2opt			GSAACPSO			MSOM			GCGA		
		Δ^*	Mean	T	Δ^*	Mean	T	Δ^*	Mean	T	Δ^*	Mean	T	Δ^*	Mean	T	Δ^*	Mean	T
eil51	426	0	426	0.2	0	426	2.7	0	426	4.9	0.23	427.27	-	1.64	435.12	0.25	0.23	430	0.98
berlin52	7542	0	7542	0.7	0	7542	4.6	0	7542	5.1	0	7542	-	0	7693.59	0.3	-	-	-
eil76	538	0	538	0.5	0	538	4.3	0	538	7.7	0	540.2	-	2.04	553.49	0.39	2.23	551	2.42
kroA100	21282	0	21282	2.4	0	21282	4.7	0	21284.33	10	0	21370.47	-	0.24	21524.61	0.53	0.05	21543	2.57
kroB100	22141	0	22159	4.2	0	22160.5	5.1	0	22160.67	10	0	22282.87	-	0.92	22528.47	0.52	0.24	22542	2.55
kroC100	20749	0	20762	2.2	0	20749	4.4	0	20749.33	10.3	0	20878.97	-	0.32	20896.32	0.5	0.32	21025	2.88
kroD100	21294	0	21315	1.2	0	21294	4.9	0	21294	9.5	0.07	21620.47	-	0.8	21536.75	0.51	1.22	21809	2.53
kroE100	22068	0	22111	2.1	0	22113.17	5.3	0	22125.2	10.1	0	22183.47	-	1.12	22507.15	0.52	0.24	22379	2.53
rd100	7910	0	7928	1.5	0	7910	4.6	0	7910	9.7	0	7987.57	-	0.99	8119.62	0.5	0.96	8031	2.55
eil101	629	0	632	3.3	0	629	5.6	0	629	10.1	0.16	635.23	-	2.07	648.81	0.51	1.59	646	2.22
lin105	14379	0	14379	1.8	0	14379	5.5	0	14382.67	11.1	0	14406.37	-	0	14427.89	0.55	0.56	14544	2.55
pr124	59030	0	59034	3.7	0	59037.67	5.6	0	59030	12.6	-	-	-	0.26	59927.26	0.62	0	59141	4.23
bier127	118282	0	118369	3.2	0	118291.5	15.2	0	118322.17	17.6	0	119421.83	-	1.25	121570.24	0.8	0.91	120412	2.52
ch130	6110	0	6153	5.4	0	6113.17	5.2	0	6119.83	12.6	0.51	6205.63	-	0.8	6282.91	0.66	-	-	-
pr136	96772	0.02	97168	8.8	0.01	96946.83	5.5	0.15	96984.17	13.1	-	-	-	0.73	99771.93	0.76	0.47	99505	3.02
pr144	58537	0	58537	5.3	0	58581.17	6.2	0	58563.5	13.8	-	-	-	-	-		0	58560	4.26
ch150	6528	0	6546	5.5	0	6550	6.3	0.32	6553.33	15.6	0	6563.7	-	1.67	6720.58	0.78	-	-	
kroA150	26524	0	26567	7.8	0	26536	6.3	0	26548.5	15.9	0	26899.2	-	1.64	27248.11	0.76	1.4	27298	3.83
kroB150	26130	0	26168	8.9	0.01	26156.67	6.7	0.02	26173.33	15.8	0	26448.33	-	0.74	26550.69	0.77	1.63	26682	3.58
pr152	73682	0	73741	5.5	0	73754.67	6.7	0	73825	16	-	-	-	1.57	75597.73	0.89	0.19	74582	4.19
d198	15780	0.14	15826	20.1	0.07	15800.33	57.8	0.07	15805.33	56.9	-	-	-	-	-	-	1.01	16084	3.78
kroA200	29368	0	29432	13.2	0.05	29498	8.5	0	29466.33	20.7	0.05	29738.73	-	1.08	30014.1	1.06	1.33	29910	4.48
kroB200	29437	0.04	29577	7.3	0	29469.33	8.1	0.03	29488.5	20.4	0.35	30035.23	-	1.82	30590.93	1.07	2.09	30627	4.55
pr226	80369	0	80399	30.1	0	80577.67	24.1	0.12	80702.67	35.3	-	-	-	-	-	-	0.35	80969	8.08
pr264	49135	0	49283	11.9	0	49135	14.8	0	49142.5	28.2	-	-	-	-	-	-	0.62	50344	8.64
pr299	48191	0.11	48562	11.2	0	48251.83	12.1	0	48225.83	31.6	-	-	-	-	-	-	4.11	50812	5.37
lin318	42029	0.3	42582	29.2	0.35	42301.83	12.5	0.28	42375.5	34.1	1.09	43002.9	-	3.63	44344.8	1.86	3.74	44191	5.48
rd400	15281	0.52	15568	69.1	0.09	15339.33	15.5	0.18	15363.33	43.2	-	-	-	-	-	-	6.11	16420	5.6
pr439	107217	0.2	108000	86.7	0.16	108039.17	27.3	0.31	108808.83	47.6	-	-	-	-	-	-	5.19	113787	4.64
rat575	6773	2.24	6973	86.1	0.27	6812.5	24.2	0.5	6822.17	60.1	1.74	6933	-	4.3	7143.48	3.62	-	-	-

statistic is 69.26 and the p-value is $4.86E^{-11}$ which strongly suggests there is at least two methods with significantly different performance [10]. The Holm procedure is used to control the family-wise error associated with multiple comparisons. The p-values, the adjusted p-values and the adjusted significance levels computed by the Holm procedure are documented online.[6] The pairwise compariosns reveal that the ADHMs achieve statistically significant improvement over GCGA, MSOM, GSAACPSO. However, the difference in the performance of the ADHMs and (AHSATS-2OPT and AHSATS-DCM) is not statistically significant.

6.2 Experiment II

The second experiment evaluates the performance of the ADHMs in comparison with the individual SMH tuned using the irace package. For each problem instance, the SMH is run for the same amount of time required for the execution of the ADHMs. The results are averaged over 30 independent runs. The percentage deviations of the mean performance of the SMHs and the ADHMs from the best known solutions are shown in Table 2 which demonstrates the superior performance of the ADHMs. The Friedman test is used to detect whether performance of the ADHMs and the SMHs is significantly different. The average rankings of

Table 2. The percentage deviation of the mean solutions from the optimal solutions.

Instance	ADHMs	SA	TS	ILS	MA	Instance	ADHMs	SA	TS	ILS	MA
eil51	0	9.35	2.13	0.52	1.22	ch150	0.28	2.74	4.99	0.72	1.24
berlin52	0	1.91	0.63	0	0.46	kroA150	0.16	2.4	5.91	0.21	1.06
eil76	0	6.4	1.12	0.35	0.45	kroB150	0.15	2.23	6.04	0.23	1.45
kroA100	0	1.13	3.65	0.01	0.48	pr152	0.08	4.86	3.02	0.18	0.76
kroB100	0.08	1.94	4.51	0.2	0.63	d198	0.29	1.54	1.45	0.23	0.97
kroC100	0.06	1.84	4.92	0.85	0.92	kroA200	0.22	3.18	7.31	0.29	1.43
kroD100	0.1	4.14	5.14	1.89	0.98	kroB200	0.48	3.77	8.07	0.69	1.79
kroE100	0.19	1.96	3.42	1.49	0.98	pr226	0.04	1.49	4.3	0.14	0.8
rd100	0.23	3.34	5.02	2.27	1.01	pr264	0.3	19.37	5.6	0.93	1.5
eil101	0.48	3.12	1.81	1.34	1.56	pr299	0.77	5.15	8.63	1.05	2.26
lin105	0	2.35	3.02	0.83	0.64	lin318	1.32	4.3	6.43	1.38	1.88
pr124	0.01	1.31	2.2	0.38	0.26	rd400	1.88	2.84	7.69	1.48	2.1
bier127	0.07	3.09	4.52	1.08	0.58	pr439	0.73	3.13	8.02	0.97	1.38
ch130	0.7	2.47	6.31	1.03	1.29	pcb442	1.75	2.91	6.06	1.87	2.4
pr136	0.41	2.64	3.47	0.8	1.39	rat575	2.95	5.96	8.93	2.5	3.48
pr144	0	2.99	2.8	0.07	0.12	-	-	-	-	-	-

[6] https://www.dropbox.com/s/0c7gtcijdlbx5w4/stat_analysis.rtf?dl=0.

the methods are ADHMs: 1.11, ILS: 2.08, MA: 2.80, SA: 4.26 and TS: 4.75. Which indicates that the ADHMs are the best. The Friedman statistic is 112.35 and the p-value is $7.18E^{-11}$ which indicates the differences are significant. The unadjusted p-values are computed from the Friedman rankings. The Holm procedure is used. The results of the statistical analysis is documented online.[7] All the pairwise comparisons are found to be significant which indicates that the ADHMs outperform each of the SMHs when used individually.

6.3 Experiment III

The third experiment tests the reusability of the evolved hybrid solvers by evaluating the performance of hybrid solvers evolved for seen instances on hidden instances. The hidden instances are divided into two classes (ClassA & ClassB) as mentioned in Sect. 5.4. For each class, a training instance is chosen and a hybrid solver is evolved for it. Then, the performance of the hybrid solver is tested on the hidden instances belonging to that class. The best and the mean performance over 30 runs are used to verify the quality of the "trained" solvers on the hidden instances. The results are shown in Table 3 in which Δ^* and Δ represent the percentage deviations from the best known solutions of the best and mean solutions respectively. In the table, `ch130` and `rd400` are the training instances.

Table 3. The percentage deviation from the best known solutions of the best and the mean performance of the hybrid solvers applied to the hidden instances.

ClassA				ClassB			
Instance	Δ^*	Δ	QL	Instance	Δ^*	Δ	QL
berlin52	0	0	0	lin318	0.1	0.83	−0.48
kroA100	0	0.01	0.01	rd400	-	-	-
kroC100	0	0.03	−0.03	fl417	0.32	0.65	0.22
ch130	-	-	-	pr439	0.17	1.03	0.6
ch150	0	0.34	0.06	pcb442	1.03	1.75	0.26
kroB150	0	0.37	0.23	d498	0.95	1.81	0.3
kroA200	0	0.35	0.13	u574	1.3	2.35	0.8
kroB200	0	0.71	0.23	rat575	1.96	2.78	0.11
pr226	0	0.21	0.18	p654	0.37	0.56	0.13
pr264	0	0.41	0.11	d657	1.32	2.11	0.31
pr299	0.02	0.68	−0.09	u724	1.33	2.32	0.27

[7] https://www.dropbox.com/s/0c7gtcijdlbx5w4/stat_analysis.rtf?dl=0.

From Table 3, the "trained" ADHM finds the best known solution for 9 instances out of 10 for ClassA. For ClassB, the deviation of best solution found by the trained solver is less than 2%. The fourth column (QL) is the quality loss which is the difference between the mean performance of the trained ADHMs and the actual ADHMs evolved for each of the hidden instances. A negative difference indicates that the trained ADHMs perform better. In general, the quality loss is less than 1% which implies that the trained ADHMs are reusable since there is not much deterioration in their performance. It is worth noting that by combining Tables 1 and 3 that the trained ADHMs perform better than GCGA, MSOM and GSAACPSO and perform slightly worse than AHSATS-DCM and AHSATS-2OPT. In particular, the difference in the mean performance of the trained ADHMs and the AHSATS-DCM (the best performer) is less than 1% for ClassA and less than 2% for ClassB (except for `rat575`). This demonstrates that the trained ADHMs are competitive to the manually crafted hybrid metaheuristics.

7 Conclusion and Future Work

The research presented in this paper is an initial attempt to ascertain the feasibility of automating the design of hybrid metaheuristics. The paper illustrates how a meta-genetic algorithm can be used to hybridize metaheuristics and dynamically tune the parameters of the individual metaheuristics. The study has revealed the potential of automating the design of metaheuristics. The evolved hybrids were found to perform competitively with manually designed hybrids and perform better than the metaheuristics applied individually. The evolved hybrid solvers were also found to be reusable. The meta-genetic algorithm is readily applicable to other domains. Technically, only three aspects need to be changed in order to apply the meta-genetic algorithm to a new problem domain: the representation of the solution; the problem-specific heuristic (the move operator) and the objective function.

Given the success in this initial study, future research will be conducted to extend this work further. In this study, the metaheuristics were combined linearly. In future work, other mechanisms for combining the metaheuristics will be investigated on different problem domains.

References

1. Adriaensen, S., Brys, T., Nowé, A.: Designing reusable metaheuristic methods: a semi-automated approach. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2969–2976. IEEE (2014)
2. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization, vol. 45. Springer Science & Business Media, Heidelebrg (2008)
3. Bhanu, S.M.S., Gopalan, N.: A hyper-heuristic approach for efficient resource scheduling in grid. Int. J. Comput. Commun. Control **3**(3), 249–258 (2008)
4. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: a survey. Appl. Soft Comput. **11**(6), 4135–4151 (2011)

5. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
6. Chen, S.M., Chien, C.Y.: Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Syst. Appl. **38**(12), 14439–14450 (2011)
7. Créput, J.C., Koukam, A.: A memetic neural network for the euclidean traveling salesman problem. Neurocomputing **72**(4), 1250–1264 (2009)
8. Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
9. Deng, W., Chen, R., He, B., Liu, Y., Yin, L., Guo, J.: A novel two-stage hybrid swarm intelligence optimization algorithm and application. Soft. Comput. **16**(10), 1707–1722 (2012)
10. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
11. Dioşan, L., Oltean, M.: Evolutionary design of evolutionary algorithms. Genet. Program Evolvable Mach. **10**(3), 263–306 (2009)
12. Durstenfeld, R.: Algorithm 235: random permutation. Commun. ACM **7**(7), 420 (1964)
13. Eiben, Á.E., Smit, S.K.: Evolutionary algorithm parameters and methods to tune them. In: Hamadi, Y., Monfroy, E., Saubion, F. (eds.) Autonomous Search, pp. 15–36. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21434-9_2
14. Gendreau, M., Potvin, J.: Handbook of Metaheuristics. International Series in Operations Research & Management Science. Springer, Heidelberg (2010)
15. Gomes, C.P., Selman, B.: Algorithm portfolios. Artif. Intell. **126**(1), 43–62 (2001)
16. Grobler, J., Engelbrecht, A.P., Kendall, G., Yadavalli, V.: Alternative hyper-heuristic strategies for multi-method global optimization. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)
17. Helsgaun, K.: An effective implementation of the lin-kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**(1), 106–130 (2000)
18. Hutter, F., Hoos, H.H., Stützle, T.: Automatic algorithm configuration based on local search. AAAI **7**, 1152–1157 (2007)
19. Kanda, J., de Carvalho, A., Hruschka, E., Soares, C., Brazdil, P.: Meta-learning to select the best meta-heuristic for the traveling salesman problem: a comparison of meta-features. Neurocomputing **205**, 393–406 (2016)
20. Lin, Y., Bian, Z., Liu, X.: Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing-tabu search algorithm to solve the symmetrical traveling salesman problem. Appl. Soft Comput. **49**, 937–952 (2016)
21. Maashi, M., Özcan, E., Kendall, G.: A multi-objective hyper-heuristic based on choice function. Expert Syst. Appl. **41**(9), 4475–4493 (2014)
22. Mühlenbein, H., Gorges-Schleuter, M., Krämer, O.: Evolution algorithms in combinatorial optimization. Parallel Comput. **7**(1), 65–85 (1988)
23. Osaba, E., Yang, X.S., Diaz, F., Lopez-Garcia, P., Carballedo, R.: An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng. Appl. Artif. Intell. **48**, 59–71 (2016)
24. Pillay, N.: Intelligent system design using hyper-heuristics. S. Afr. Comput. J. **56**(1), 107–119 (2015)
25. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976)

26. Saenphon, T., Phimoltares, S., Lursinsap, C.: Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. Eng. Appl. Artif. Intell. **35**, 324–334 (2014)
27. Talbi, E.G.: A taxonomy of hybrid metaheuristics. J. Heuristics **8**(5), 541–564 (2002)
28. Wu, C., Liang, Y., Lee, H.P., Lu, C.: Generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining. Phys. Rev. E **70**(1), 016701 (2004)