Grammatical Rules for the Automated Construction of Heuristics

Germán Terrazas Natalio Krasnogor

Abstract— **Developing a problem-domain independent methodology to automatically generate high performing solving strategies for specific problems is one of the challenging trends on hyper-heuristics design. Designing hyper-heuristics is important because they raise the level of generality on automated problem solving by attempting to select and/or generate tailored heuristics for the problem at hand. In this paper, we present a three-steps methodology that combines multiple sequence alignment and grammatical induction in order to automatically generate high performing solving strategies for a combinatorial optimisation problem. We present proof-of-concept results of applying this methodology to instances of the well-known symmetric TSP. The goal here is to demonstrate *feasibility* rather than compete with state of the art TSP solvers. This TSP is chosen only because it is an easy to state and well known problem.**

I. Introduction

A *hyper-heuristic* is a search methodology that selects and combines heuristics to generate good solutions for a given problem. Studying hyper-heuristics is important because they provide a problem-independent level of abstraction for the automatic generation of good performing algorithms. In our previous work [1], we reported on a methodology for the automated manufacture of heuristic search strategies that produces good solutions for a given combinatorial optimisation problem. Such methodology focused on the identification and combination of beneficial cooperative structures across three consecutive stages: pattern-based heuristics generation, cross validation and template-based heuristics distilling. In particular, the last stage of that approach involved the manual specification of a template from where effective and efficient heuristics are drawn. Here, we present an alternative to such process where the structure generalisation offered by grammatical induction algorithms is employed to obtain a set of rules to, subsequently, derive high performing heuristics. Therefore, this paper brings an improved method for the automated construction of heuristic search strategies combining multiple sequence alignment and grammatical induction across three main stages: pattern-based heuristics generation, cross validation and grammar-based heuristics construction. The following section gives a brief introduction to hyper-heuristics and the context of our investigation. Section III reviews some of the applications of grammar induction algorithms and presents the grammatical induction algorithm employed here. Section IV expands on the proposed approach giving details of the model components and the methodology. After that, experiments and results are presented and discussed in Section V. Finally, conclusions and further work are the subject of Section VI.

Germán Terrazas, ASAP Group, School of Computer Science, University of Nottingham, UK, email:gzt@cs.nott.ac.uk

Natalio Krasnogor, ASAP Group, School of Computer Science, University of Nottingham, UK, email:nxk@cs.nott.ac.uk

II. Heuristics Design

Hyper-heuristics are defined as search methodologies that select and combine low-level heuristics to solve hard computational search problems [2]. The general aim of a hyper-heuristic is to manufacture unknown heuristics which are fast, well performing and widely applicable to a range of problems. During the fabrication process, hyper-heuristics receive feedback from the problem domain which indicates how good are the chosen heuristics for solving the problem at hand, hence driving the search process. Studying novel approaches for the development of hyper-heuristics is important since they are domain-independent problem strategies that operate on a space of heuristics, rather than on a space of solutions, and rise the level of generality on automated problem solving. Hyper-heuristics have been employed for solving search and optimisation problems such as timetabling [3], scheduling [4][5] and satisfiability [6] among others. Recent investigations on hyper-heuristics have sprung up in two main different directions of hyper-heuristics: 1) heuristics that choose heuristics and 2) heuristics that generate heuristics. In the first case, a learning mechanism assists the selection of low-level heuristics according to their historical performance during the search process, e.g. [5]. In the second case, the focus is on searching components that once combined generate a new heuristic suitable for the problem at hand. For example, approaches based on genetic algorithms [4] and genetic programming have been proposed for the automated generation of heuristics [7]. From an engineering point of view, the already existent approaches are defined in terms of an architecture established by the underlying meta-heuristic which sometimes brings unsuspected difficulties such as the correct modelling of solutions or parameters tunning. Employing a process that learns how to build adequate strategies [8][9] is another path to explore, e.g. fractal instances were used before to learn TSP rules based on learning classifier systems in [10]. We consider that employing a method that takes as input high performing heuristics, learns about its features, and returns a meaningful description of such to manufacture others with similar structural features and performance is an alternative route within the second flavour of hyper-heuristics. Hence, our interest here lays on whether the use of a grammatical induction algorithm would be able to infer a grammar for deriving heuristics that construct high quality solutions when applied to the problem at hand.

III. Applications of Grammatical Inference

Grammatical inference can be seen as a reverse engineering process that attempts to construct a grammar by means of generalisation of the underlying structure of a given set of words of a certain language. Studies on grammatical inference methods currently classify the existing approaches according to the type of input, i.e. tag-based methods and word-based methods [11]. Despite the open problems and key issues on grammatical inference [12], many are the algorithms and techniques successfully employed for constructing grammars across different research fields. For instance, in [13] the authors present the application of grammar induction to model job traffic in a grid. In particular, the proposed approach employs a deterministic finite automata that induces and combines traffic patterns occurring across a grid infrastructure in order to facilitate its management support. Another application is reported in [14] where a grammatical inference method is proposed to derive a grammar from a surveillance system. In this case, the aim is to infer a grammar that captures the behaviour of cars when entering to, parking in and leaving from a parking lot. Grammars have also been applied to complex sequence analysis such as in [15] where a context-sensitive deterministic grammar is employed to parse DNA structures in order to spot antibiotic resistance genes.

The automatic distillation of structure (ADIOS) [16] is a word-based and unsupervised learning algorithm that extracts syntactic rules from a given input set of sentences (sequences of words) called corpus. The algorithm comprises three main stages: initialisation, pattern distillation and generalisation. During the first stage, each sentence is loaded on a graph, the nodes of which are formed by the words and the path across the nodes map the sentence. Afterwards, pattern distillation and generalisation are iteratively applied on each path. In the former, significant patterns of words are identified whilst the construction of equivalent classes of words is performed in the later. This process gives as a result a hierarchical arrangement of patterns and equivalent classes that is interpreted as a grammar. Fig. 1 shows a corpus example, the hierarchical arrangement of patterns and equivalent classes, and the induced grammar resulting from ADIOS.

IV. Proposed Approach

In [1] we reported on a methodology capable of manufacturing heuristic search strategies as the result of the consecutive application of three steps: pattern-based heuristics generation, cross validation and template-based heuristics distilling. In particular, the goal of the last part of such methodology was to discover efficient building blocks of low-level heuristics that give rise to a template from where a family of better than average heuristics could be drawn. Such process involved the manual construction of templates in terms of cooperative low-level heuristics resulting from an multiple sequence alignment process. In order to automatise this process, we present here a faster, more general and less human-dependent alternative to generate the family

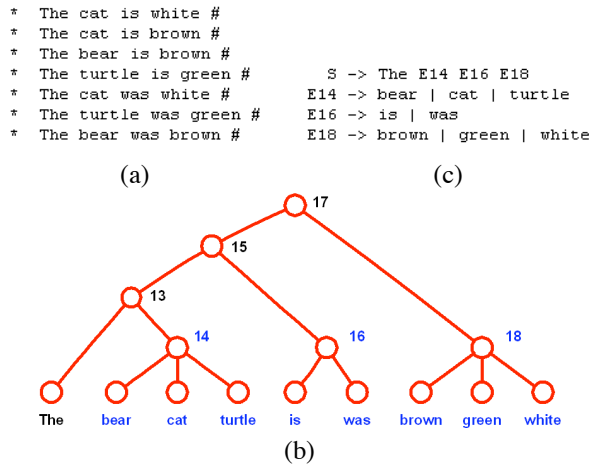

Fig. 1. Input set of sentences (a), patterns and equivalence classes (b) and induced grammar (c).

of good performing heuristics. As we aim to produce a complete picture of our research, what follows provides a full introduction to our methodology including the goals and descriptions of the first two stages brought from our previous work.

Given a set of instances of a combinatorial optimisation problem Π, we propose a methodology composed of pattern-based heuristics generation, cross validation and grammar-based heuristics construction. Each stage is associated to a dataset generated from the optimisation problem at hand whilst the output of the methodology is a grammar to be employed for the manufacture of good performing heuristics. Fig. 2 depicts the methodology and its components.

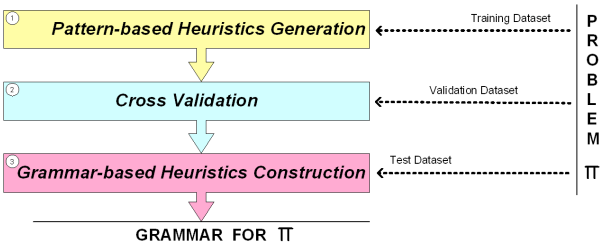

Fig. 2. Schematic representation of the proposed methodology with its three stages, their associated datasets and the achieved template for the problem at hand.

In the *pattern-based heuristics generation*, an input dataset is employed to train randomly generated sequences of low-level heuristics (high-level heuristics). This training aims at generating proficient high-level heuristics, the common constituents of which are expected to produce high quality solutions when applied to a given instance of the problem at hand. The research question in this stage is:

Given a set of high-level heuristics, is it possible to generate common combinations of low-level heuristics ? If yes, how do they look like and how reliable are these combinations ?

In order to address the first question, a process that spots common combinations of low-level heuristics (patterns) and constructs pattern-based heuristic is employed. The goal of the *cross validation* is then to assess the performance of the constructed pattern-based heuristic over a validation dataset comprising similar instances of the combinatorial optimisation problem at hand. Thus, the question in this stage is:

What is the performance of a pattern-based heuristic when applied to a set of different problem instances ?

The goal of the *grammar-based heuristics construction* stage is to obtain a grammar from where better than average high-level heuristics can be manufactured. Thus, the constructed pattern-based heuristics form the input to the ADIOS algorithm, the output set of rules of which is expected to derive high-performing heuristics. Here, an extra dataset is employed to test the performance of the derived heuristics. The question in this stage is:

Can pattern-based heuristics be characterised by a grammar ? If yes, what is the performance of the heuristics generated with that grammar when applied to a set of different problem instances ?

The above methodology is expected to deliver a procedure for the automated construction of efficient heuristic search strategies. In order to produce a self-contained report, the following section revisits the experiments and results of the pattern-based heuristics generation stage and cross validation stage of our methodology reported in [1].

V. Methods and Results

This section presents the findings obtained by the above methodology. The chosen combinatorial optimisation problem is the widely known symmetric Traveling Salesman Problem (TSP). The TSP instance considered here is kroA100 which comprises 100 cities distributed in the Euclidean space. The objective value corresponding to the known optimum solution (shortest tour) for this instance is 21282 (see TSPLIB[1]). For each stage of our methodology, we generated five sets in the following systematic way. Each set is initialised with ten copies of the known optimum solution for kroA100. Each of this initial solutions is then 'disturbed' with n consecutive city swaps. In this way, setting n to 5, 25, 50, 75 and 100, a total of ten independently 'disturbed' tours per set are obtained.

We consider a high-level heuristic to consist of a sequence of two or more low-level heuristics with or without repetition. For the chosen problem, a low-level heuristic is a local search for the TSP that could be deterministic (e.g. always selecting the best of a set of improving two-edges interchange) or stochastic (e.g. selecting at random from a set of improving two-edges interchange, and hence potentially giving different results if re-executed). Eight low-level heuristics were implemented: *2-opt*, *3-opt*, *OR-opt* and *node insertion* which are deterministic, and *1-city insertion*, *2-exchange*, *arbitrary insertion* and *inver-over* which are stochastic. These eight low-level heuristics are originally defined in [17][18][19][20][21].

[1]http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

A. Pattern-based Heuristics Generation

1) Training Datasets: In this stage, each of the perturbed tours, labeled as $tkroA100_i^n$, i=0 . . . 9, n=5, 25, 50, 75, 100, is independently considered for training. A sample of the training data, grouped by set (n), is listed in Table I where the values indicate the percentage distance to the optimum from each perturbed tours.

TABLE I
SAMPLE PERTURBED TOURS OF THE TRAINING DATA SET

Set	$tkroA100_0^n$	$tkroA100_1^n$	$tkroA100_2^n$
$n = 5$	1.42669	1.27600	1.79926
$n = 25$	4.25805	4.60262	4.13631
$n = 50$	6.39869	6.46067	5.76585
$n = 75$	7.01362	6.38215	6.75190
$n = 100$	6.80147	6.59012	6.93252

2) Method: For a given disturbed tour ($tkroA100_i^n$), a set containing 500 high-level heuristics generated at random was created. Then, each of the 500 high-level heuristics was independently applied to the associated perturbed tour. In this context, an application is seen as a pipeline process in which the chain of processing elements is given by the sequence of low-level heuristics and the information to be processed is the disturbed tour. Thus, the low-level heuristics are applied one after another in the order in which they appear in the sequence and producing better or equal solutions at each step, i.e. operating in a hill climber style [22]. To illustrate this process, Fig. 3 depicts how a high-level heuristic comprising two 2-exchange and a 1-city insertion is applied to a TSP instance.

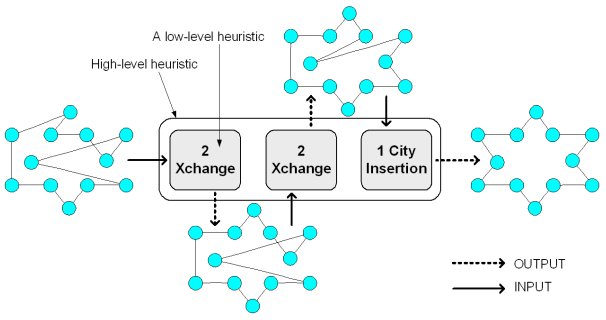

Fig. 3. A high-level heuristic in which successive applications of two 2-exchange and a 1-city insertion low-level heuristics find the optimum solution for the Star of David tour [23][24].

In order to identify common combinations of low-level heuristics, the 500 high-level heuristics are then sorted according to the distance between the solution that their

applications produce and the known optimum solution. The top five high-level heuristics are then selected and encoded as sequences of characters using 'A' to represent 1-city insertion, 'C' to represent 2-opt, 'D' to represent 3-opt, 'E' to represent OR-opt, 'T' to represent 2-exchange, 'F' to represent node insertion, 'G' to represent arbitrary insertion and 'H' to represent inver-over. Hence, in order to identify common combinations of low-level heuristics among the filtered sequence, we employ a multiple sequence alignment (MSA) method [25] over the encodings. For instance, Fig. 4 highlights in gray the common combinations found among the best five high-level heuristics generated for $tkroA100_2^{75}$.

```
1-HLH^75_3   ehhG--thHG------HHtG---------thhD---DhDh
2-HLH^75_3   ----aD---GdadtdtH---------------DddCD-D-
3-HLH^75_3   -----D--HG------H--Gaccchccaca--D--C----
4-HLH^75_3   ---G----HG------HH-G------------D-------
5-HLH^75_3   ---Ge----G------H--G------------D---D---
                |  |   ||       ||  |              |   ||  |
                |  |   ||       ||  |              |   ||  |
PBH^75_3     ---G-D--HG------HH-G------------D--CD-D-
```

Fig. 4. Multiple sequence alignment of the top five heuristics. Capitals highlighted in gray indicate the common sequences of heuristics.

The results obtained by the MSA method reveal that there are indeed occurrences of common combinations, i.e. patterns of low-level heuristics, among the best ranked high-level heuristics. Thus, these findings give a positive answer to the research question stated for the first part of our methodology in Section IV.

From the resulting alignment, we construct a consensus sequence capturing and representing regions of similarity. We define this consensus sequence as a pattern-based heuristic (PBH_i^n) associated to a perturbed tour ($tkroA100_i^n$). The constructing procedure consists in copying the matching characters between two or more encodings into a new sequence from left to right and following the position in which they appear. For instance, Fig. 4 shows that PBH_3^{75} is the resulting pattern-based heuristic encoded as GDHGH-HGDCDD, after combining the common patterns from the high-level heuristics 1-HLH_3^{75} to 5-HLH_3^{75}. Given that this new heuristic is built in terms of common combinations of low-level heuristics, its performance is then expected to be as good as (or better than) any of the top ranked. Notice that the length of the constructed heuristic varies according to the number of matches. Since this is related to the way in which the construction procedure is defined, alternative methodologies to obtain the optimal common sequence are open to further investigation.

In order to assess the reliability of the identified patterns, we then proceed to evaluate the performance of PBH_i^n against a set of high-level heuristics (different than the initial ones) with the hope that, on average, the best tour improvements are obtained by the former. In order to do this, 300 copies of PBH_i^n are obtained and for each of them a new high-level heuristic equal in length is randomly created. Each of these heuristics is then independently applied to $tkroA100_i^n$ a total

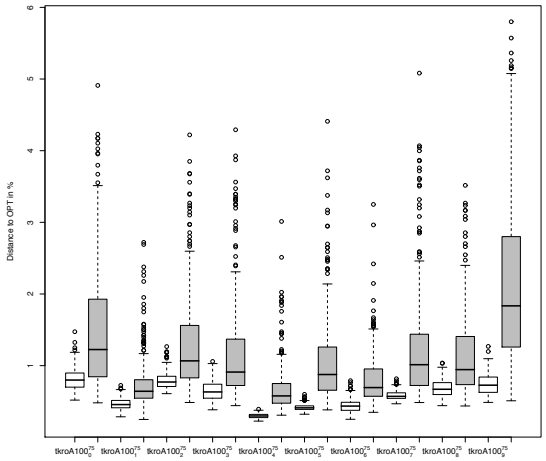

Fig. 5. Assessment of ten pattern-based heuristics resulting from independent sequence alignments. Each pair of boxplots summarises a vis-a-vis comparison between the performance of 300 copies of PBH_i^{75} and the performance of other 300 high-level heuristics when applied to $tkroA100_i^{75}$ for i=0 . . . 9. The horizontal lines in the boxes indicate the median, upper (lower) end of the whiskers the maximum (minimum) whilst the outside points are outliers.

of 10 times and the average percentage distance between the lengths of the resulting tours and the known optimum is considered as the measure of their performance. As an example, Fig. 5 shows the assessment of the 10 pattern-based heuristics obtained from the data set generated with n=75.

According to the results, it is clear that the performance of pattern-based heuristics (white boxplots) is better in average than the performance of the non-pattern-based high-level heuristics (gray boxplots). These findings constitute a positive answer to the second research question stated in the first stage of the presented methodology, i.e. the identified common-sequences of heuristics are indeed reliable.

B. Cross Validation

1) Validation Dataset: The cross validation data are given in sets of ten perturbed tours $vkroA100_i^n$, i=0 . . . 9. A sample of the data, grouped by set (n), is listed in Table II where the values indicate the percentage distance to the optimum from each perturbed tours.

TABLE II
SAMPLE PERTURBED TOURS OF THE VALIDATION DATA SET

Set	$vkroA100_0^n$	$vkroA100_1^n$	$vkroA100_2^n$
$n = 5$	1.86490	1.72394	1.41001
$n = 25$	5.38403	5.42246	3.76134
$n = 50$	6.85800	6.13800	6.66469
$n = 75$	6.92453	6.57452	6.85969
$n = 100$	7.58471	6.69500	6.90264

2) Method: The goal of this stage is to perform a cross validation analysis in order to assess the performance of the pattern-based heuristics over a set of disturbed tours. Thus, for each combination of PBH_j^n and $vkroA100_i^n$, i, j=0 . . . 9, a total of 300 copies of PBH_j^n are obtained and, for each

of the copies, a new high-level heuristic equal in length is randomly created. Then, the heuristics are independently applied to the given $vkroA100_i^n$ a total of 10 independent times and the average percentage distance between the lengths of the resulting tours and the known optimum is considered as the measure of their performance. Fig. 6 shows the resulting assessment of a pattern-based heuristic, encoded as GDHGHHGDCDD, over the 10 perturbed tours belonging to the data set generated with n=75.

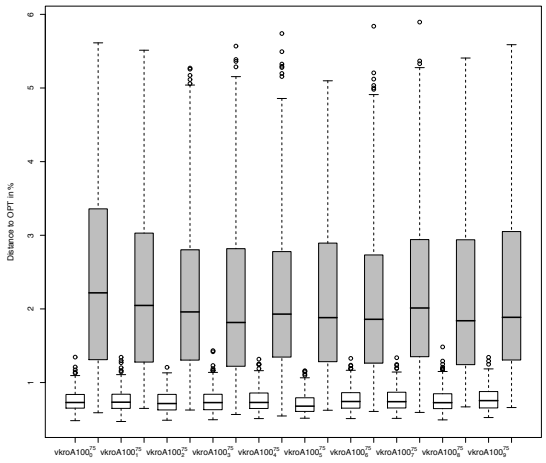

Fig. 6. Performance evaluation of a pattern-based heuristic across the perturbed tours belonging to the data set generated with n=75. Each pair of boxplots summarises a vis-a-vis comparison between the performance of 300 copies of GDHGHHGDCDD and the performance of other 300 high-level heuristics when applied to $vkroA100_i^{75}$. The horizontal lines in the boxes indicate the median, upper (lower) end of the whiskers the maximum (minimum) whilst the outside points are outliers.

Clearly, the performances of pattern-based heuristics (white boxplots) are better in average than the performance of the ones generated for assessment (gray boxplots). These findings answer the research question estated in the second part of Section IV, revealing that a pattern-based heuristic is in general well performing when applied to a set of different problem instances. In addition, the similar level of performance observed among the white boxplots gives an indication that common low-level heuristics could be acting as building blocks among the PBH_j^n, j=1 . . . 10.

C. Grammar-based Heuristics Construction

1) Test Dataset: The data used in this last stage comprise five sets listed in Table III. Thus, for a given experiment, each of the ten perturbed tours $gkroA100_i^n$, i=0 . . . 9, is independently employed for testing.

TABLE III
SAMPLE PERTURBED TOURS OF THE TEST DATA SET

Set	$vkroA100_0^n$	$vkroA100_1^n$	$vkroA100_2^n$
$n = 5$	1.43750	1.12729	0.80584
$n = 25$	4.00032	4.70731	4.01320
$n = 50$	5.61831	6.44469	5.96786
$n = 75$	6.34000	6.28794	6.57973
$n = 100$	6.86100	6.69199	7.10008

2) Method: The goal of this stage is to obtain a grammar capable of generating encodings of new high-level heuristics, the performances of which are expected to be in average similar or better than the pattern-based heuristics. It is here where we employ the ADIOS algorithm which through the application of its three-steps procedure, i.e. initialisation, pattern distillation and generalisation, is expected to generalise the underlying structures of the pattern-based heuristics and, hopefully, induce the most appropriate grammar. Thus, for a given n, we consider all the PBH_i^n, i=0 . . . 9, as input corpus to the ADIOS algorithm. In order to comply to the input format of ADIOS, i.e. sequence of words, each PBH_i^n is converted into a sequence of blank separated characters followed by underscores, e.g. PBH_3^{75}=CDHGHHGDCDD becomes C_ D_ H_ G_ H_ H_ G_ D_ C_ D_ D_. Considering n=75, Fig. 7 presents a sample corpus for the ADIOS algorithm, Fig 8 depicts the resulting hierarchical aggregation of patterns and equivalence classes, and Fig. 9 shows the induced grammar for that corpus.

```
*  G_ D_ H_ C_ T_ G_ A_ T_ H_ G_ T_ C_ C_ D_ H_ A_ H_ D_ A_ #

*  F_ G_ D_ A_ A_ G_ H_ G_ D_ H_ G_ C_ H_ D_ H_ D_ H_ D_ C_ H_ C_ #

*  G_ D_ H_ G_ H_ H_ G_ D_ C_ D_ D_ #

*  G_ D_ C_ C_ A_ H_ G_ H_ G_ C_ D_ C_ T_ C_ A_ C_ G_ G_ T_ #

*  C_ A_ D_ C_ F_ T_ A_ G_ D_ D_ D_ C_ G_ D_ D_ A_ D_ #

*  G_ A_ G_ C_ G_ A_ C_ H_ A_ D_ A_ C_ D_ A_ C_ A_ D_ A_ H_ C_ D_
   T_ A_ D_ A_ #

*  D_ C_ D_ G_ C_ A_ G_ A_ D_ D_ C_ C_ D_ A_ H_ H_ D_ C_ D_ D_ #

*  C_ H_ H_ H_ F_ F_ G_ H_ T_ G_ A_ G_ T_ C_ H_ C_ #

*  C_ C_ T_ G_ T_ D_ C_ C_ C_ D_ T_ C_ A_ C_ D_ G_ C_ D_ A_ D_ C_
   C_ C_ D_ D_ C_ C_ C_ #

*  F_ F_ D_ D_ D_ G_ A_ D_ H_ G_ C_ G_ A_ C_ D_ D_ H_ H_ C_ H_ C_
   C_ H_ C_ H_ D_ C_ A_ D_ D_ #
```

Fig. 7. A corpus for the ADIOS algorithm comprising the converted PBH_i^{75}, i=0 . . . 9. Each of the sequences encodes a pattern-based heuristic obtained from the first stage of our methodology.

```
   S -> P18 P20

P11 -> D_ E12
P13 -> C_ E14
P15 -> G_ E12
P16 -> E17 E14
P18 -> P16 E19
P20 -> E21 E22

E12 -> A_ | D_
E14 -> A_ | C_ | D_ | H_ | T_ | P11
E17 -> A_ | D_ | H_ | T_ | P11 | P13 | P15
E19 -> D_ | P16
E21 -> C_ | G_ | P13 | P16
E22 -> H_ | T_ | P11 | P13 | P15
```

Fig. 9. The grammar Gen^{75} inferred from the input set PBH_i^{75}, i=0 . . . 9, where S is the start symbol and non-terminals starting with P and E denote patterns and equivalence classes respectively.

After the ADIOS algorithm is ran, the inferred grammar GGen^n is employed to derive 5000 unique encodings of new high-level heuristics, i.e. grammar generated heuristics. In order to assess the reliability of these new high-level heuristics, we compared their performance against randomly generated high-level heuristics expecting that, on average,

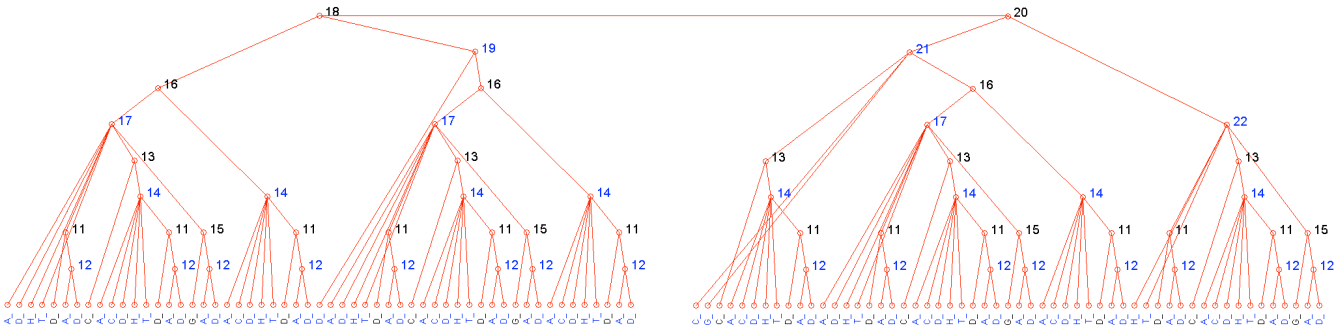

Fig. 8. Hierarchy of patterns (red labels) and equivalence classes (blue labels) resulting from the corpus shown in Fig. 7.

the best tour improvements are obtained by the former. Thus, for each $gkroA100_i^n$, a total of 300 different grammar generated heuristics are randomly chosen and, for each of these, an associated high-level heuristic is randomly created for assessment. In particular, these two heuristics are equal in length and with at least 70 per cent of dissimilarity in structure. Each grammar generated heuristic and its corresponding randomly created one are then independently applied to copies of the given $gkroA100_i^n$ a total of 10 independent times. At the end of the evaluations, the average distance between the lengths of the resulting tours and the known optimum is considered as the measure of performance. The resulting assessments for n=5, 25, 50, 75, 100 is presented in Fig. 10 where, for each of the $gkroA100_i^n$, the boxplots show grammar generated heuristics assessed against their associated randomly generated ones.

On the one hand, the results of this stage demonstrate that it is possible to characterise the pattern-based heuristics with a grammar which is capable of deriving encodings of well performing high-level heuristics. This fact constitutes a positive answer for the first question established in the third part of our methodology. Overall, the resulting assessments in Fig. 10 show that the performances of the grammar-based heuristics (white boxplots) are on average better than the performances of the randomly generated high-level heuristics (gray boxplots). In addition, considering the boxplots of the pattern-based heuristics (Fig. 6) and the boxplots of the grammar-based heuristics (Fig. 10), it is also shown that both performances are in general similar. This last fact comes as a consequence that their encodings are expected to be part of the same language and, hence, display an operative resemblance when applied to the problem at hand.

On the other hand, considering the standard deviations and maximum values of Fig. 10 (d), it is evident that some of the randomly generated heuristics have bigger opportunity to outperform the ones derived from the grammar. Naturally, one of the reasons for this is that during the random generation, appropriate combinations of low-level heuristics with more efficient local interactions could be generated (by chance). Also, there is a possibility that some of the encodings of the randomly generated heuristics are actually part of the language generated by the induced grammar. Hence, it is open to further investigation to know whether the ADIOS algorithm has been able to properly generalise the given corpus in order to identify the most appropriate grammar from the given set of encodings. For instance, the performance of the outliers shown in Fig. 10 (c) and Fig. 10 (e) may indicate that some of the grammar-generated heuristics are structurally equivalent to randomly generated encodings. This conjecture is also supported by analyzing the bias on the performance between randomly generated heuristics and grammar-generated heuristics for some $gkroA100_i^n$. Therefore, it could be the case that different combinations of input parameter values of the grammatical induction algorithm may be needed to fine tune the level of abstraction and, hence, give as a result a more reliable set of rules that generates less biased and more proficient high-level heuristics. Additionally, the number and length of the sequences of the corpus also play a very important role during the pattern distillation and generalisation phase of the ADIOS algorithm. In other words, the richer the corpus is initially set to the method, the more information is available for generalisation, hence, influencing on the quality of the acquired patterns and equivalence classes.

All in all, the results achieved in this part of the methodology show that the grammar-based heuristics construction brings indeed a more general, faster, and less human-dependent way to automatically manufacture high performing heuristic strategies to solve the problem at hand. Overall, the outcome of the assessments answers the last question of the proposed methodology. That is, the high-level heuristics derived from grammars are, in general, well performing when applied to any disturbed tour of a given data set.

VI. Conclusions

In this paper, we proposed an approach for the automated design of heuristics following the rationale of hyper-heuristics as methods to generate tailored heuristics for a given combinatorial optimisation problem. Our methodology unfolds across three stages: pattern-based heuristics construction, cross validation and grammar-based heuristics construction. As a proof of concept, we have applied our approach to instances of the symmetric TSP. As a result, the findings of the pattern-based heuristics construction stage confirmed that there are indeed common patterns of low-level heuristics among the top ranked high-level heuristics. In order to assess

the reliability of these recurrent structures, pattern-based heuristics were afterwards constructed and cross validated against randomly generated heuristics. The results achieved during the second stage proved that the spotted patterns are local search strategies beneficial to achieve good solutions when solving a symmetric TSP instance. These constructed pattern-based heuristics served as input to the grammar-based heuristics construction stage. The results of this revealed that it is possible to induce a grammar from a set of pattern-based heuristics and that the derived high-level heuristics of such grammar give, in general, high quality solutions when applied to the perturbed tours. In other words, the last part of our approach has resulted in a formal specification to automatically create heuristics that produce high quality solutions when applied to the problem at hand.

From a functional point of view, the patterns-based heuristics achieved in the methodology can be seen as combinations of beneficial structures needed for the manufacturing of high quality solutions. These combinations seemed to shared a common underlying structure that the employed grammatical induction algorithm was able to capture and generalise giving as a result hierarchical aggregations of patterns and equivalence classes. The transcription of these hierarchy of aggregates resulted in a grammar, the rules of which allow us to derive encodings of new well performing high-level heuristics. The benefits of applying our methodology lays on the use of a powerful generalisation mechanism brought in by the grammatical induction algorithm. This key component focuses on the structure of the constructed pattern-based heuristics (syntax) without having any knowledge of their performance for solving problems (semantics) during the grammar induction process.

To continue with our methodology, future work involves the extension of our approach to other instances of TSP as well as to different combinatorial optimisation problems. In addition, we consider to explore different settings of the employed grammar induction algorithm since, as discussed before, there are few cases in which the derived grammar-based heuristics are outperformed by the randomly generated ones. Moreover, besides the ADIOS algorithm is successfully deriving new encodings from the generated grammar, there are exceptional cases where unspecified terminals are employed. Evidence of this is given by the induced grammar shown in Fig. 7 where the 'F' character appearing in four of the sequences of Fig. 9 has been missed, but employed in the production of grammar-based heuristics.

ACKNOWLEDGMENT

We would like to thanks Dr Dario Landa-Silva for numerous suggestions and corrections that have helped us improve this paper. The research reported here is funded by EPSRC grant EP/D061571/1 Next Generation Decision Support: Automating the Heuristic Design Process.

REFERENCES

[1] G. Terrazas, D. Landa-Silva, and N. Krasnogor, "Discovering beneficial cooperative structures for the automated construction of heuristics," in *Stud. in Comp. Intel.* Springer-Verlag, 2010, pp. 89–100.

[2] P. Ross, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support*. Springer, 2005, ch. Hyper-heuristics, pp. 529–556.

[3] N. Pillay and W. Banzhaf, "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem," *European Journal of Operational Research*, vol. 197, no. 2, pp. 482–491, 2009.

[4] P. Cowling, G. Kendall, and L. Han, "An investigation of a hyper-heuristic genetic algorithm applied to a trainer scheduling problem," in *IEEE Congress on Evolutionary Computation*. IEEE Computer Society, 2002, pp. 1185–1190.

[5] P. Cowling and K. Chakhlevitch, "Hyperheuristics for managing a large collection of low level heuristics to schedule personnel," in *IEEE Congress on Evolutionary Computation*. IEEE Computer Society, 2003, pp. 1214–1221.

[6] M. Bader-El-Den and R. Poli, "A gp-based hyper-heuristic framework for evolving 3-sat heuristics," in *Genetic and Evolutionary Computation Conference*. ACM, 2007, pp. 1749–1749.

[7] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one," in *Genetic and Evolutionary Computation Conference*. ACM, 2007, pp. 1559–1565.

[8] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[9] N. Krasnogor, *Handbook of Natural Computation*. Springer Berlin / Heidelberg, 2009, ch. Memetic Algorithms, p. (to appear).

[10] M. Tabacman, N. Krasnogor, J. Bacardit, and I. Loiseau, "Learning classifier systems for optimisation problems: a case study on fractal travelling salesman problem," in *Genetic and Evolutionary Computation*. ACM, 2008, pp. 2039–2046.

[11] B. Cramer, "Limitations of current grammar induction algorithms," in *45th Annual Meeting of the ACL*. Morristown, NJ, USA: Association for Computational Linguistics, 2007, pp. 43–48.

[12] C. D. L. Higuera, "Current trends in grammatical inference," in *Joint IAPR International Workshops on Advances in Pattern Recognition*. Springer-Verlag, 2000, pp. 28–31.

[13] W. Mulder and C. Jacobs, "Grid management support by means of collaborative learning agents," in *6th Grids Meets Autonomic Computing*. ACM, 2009, pp. 43–50.

[14] S. Geyik and B. Szymanski, "Event recognition in sensor networks by means of grammatical inference," in *INFOCOM 2009, IEEE*, April 2009, pp. 900–908.

[15] G. Tsafnat, E. Coiera, S. Partridge, J. Schaeffer, and J. Iredell, "Context-driven discovery of gene cassettes in mobile integrons using a computational grammar," *BMC Bioinformatics*, vol. 10, no. 1, p. 281, 2009.

[16] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised context sensitive language acquisition from a large corpus," in *NIPS*, 2003.

[17] G. Babin, S. Deneault, and G. Laporte, "Improvements to the or-opt heuristic for the symmetric traveling salesman problem," *Journal of the Operational Research Society*, no. 58, pp. 402–407, 2007.

[18] J. Brest and J. Zerovnik, "A heuristic for the asymmetric traveling salesman problem," in *6th Metaheuristics International Conference*, 2005, pp. 145–150.

[19] N. Krasnogor and J. Smith, "Memetic algorithms: The polynomial local search complexity theory perspective," *Journal of Mathematical Modelling and Algorithms*, vol. 7, pp. 3–24, 2008.

[20] G. Reinelt, *The traveling salesman: Computational solutions for TSP applications*. Springer-Verlag, 1994.

[21] G. Tao and Z. Michalewicz, "Inver-over operator for the tsp," in *5th International Conference on PPSN*. Springer-Verlag, 1998, pp. 803–812.

[22] E. Özcan, B. Bilgin, and E. Korkmaz, "Hill climbers and mutational heuristics in hyperheuristics," in *9th International Conference on PPSN*, 2006, pp. 202–211.

[23] A. Mariano, P. Moscato, and M. Norman, "Arbitrarily large planar etsp instances with known optimal tours," *Pesquisa Operacional*, vol. 15, pp. 89–96, 1995.

[24] P. Moscato and M. Norman, "An analysis of the performance of traveling salesman heuristics on infinite-size fractal instances in the euclidean plane," *ORSA Journal on Computing*, 1994.

[25] J. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*. PWS Publishing, 1997.

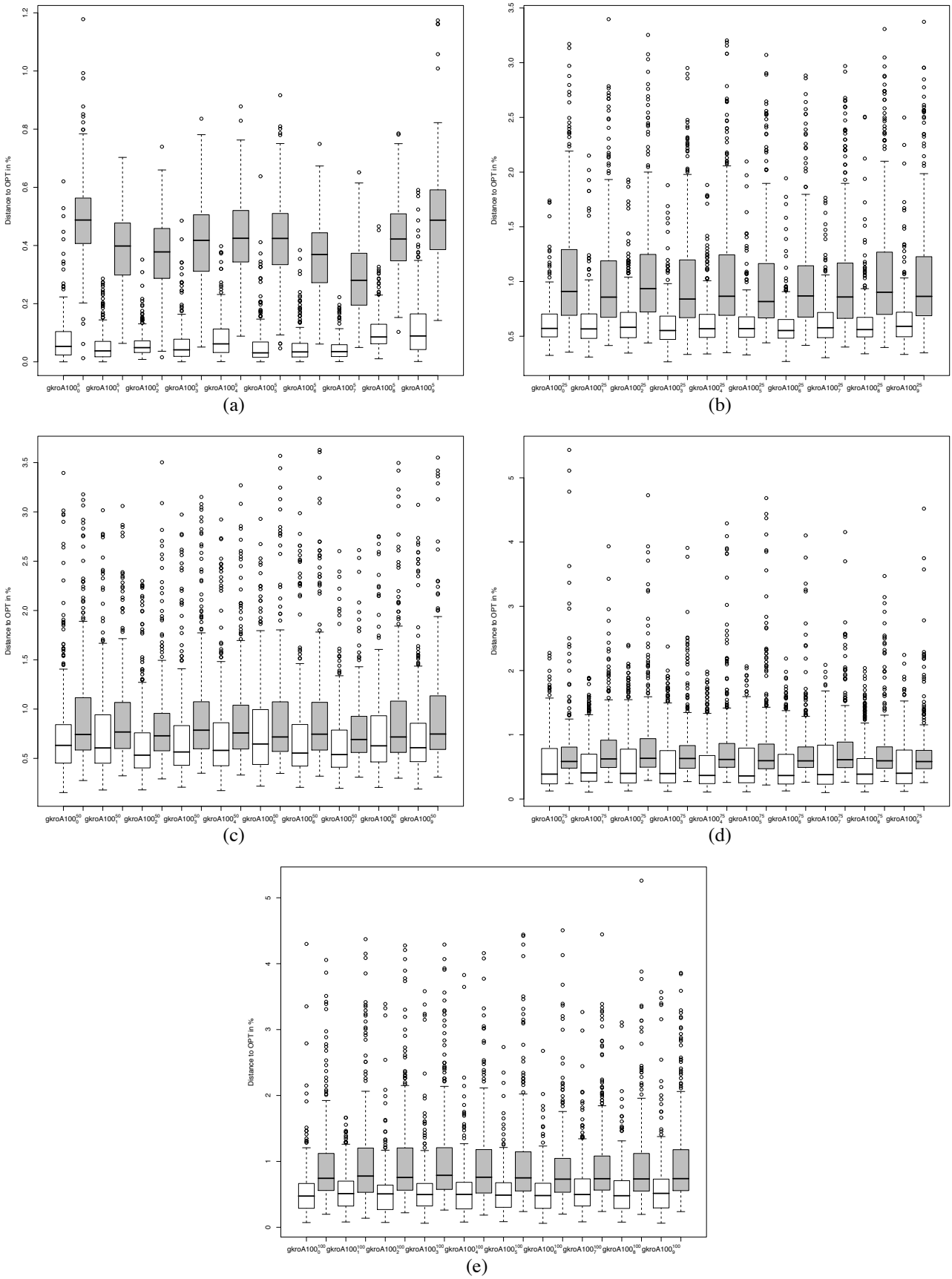

Fig. 10. Performance evaluation of grammar generated heuristics applied to perturbed tours generated with n=5, 25, 50, 75, 100. Each pair of boxplots summarises a vis-a-vis comparison between the performance of 300 heuristic generated from GGen^n and the performance of other 300 randomly generated high-level heuristics when applied to $gkroA100_i^n$ for i=0 . . . 9. The horizontal lines in the boxes indicate the median, upper (lower) end of the whiskers the maximum (minimum) whilst the outside points are outliers.