Multi-Level Grammar Genetic Programming for Scheduling in Heterogeneous Networks

Takfarinas Saber[1(✉)], David Fagan[1], David Lynch[1], Stepan Kucera[2], Holger Claussen[2], and Michael O'Neill[1]

[1] Natural Computing Research and Applications Group, School of Business, University College Dublin, Ireland
{takfarinas.saber, david.fagan, m.oneill}@ucd.ie, david.lynch@ucdconnect.ie
[2] Bell Laboratories, Nokia, Dublin, Ireland
{stepan.kucera, holger.claussen}@nokia-bell-labs.com

Abstract. Co-ordination of Inter-Cell Interference through scheduling enables telecommunication companies to better exploit their Heterogeneous Networks. However, it requires from these entities to implement an effective scheduling algorithm. The state-of-the-art for the scheduling in Heterogeneous Networks is a Grammar-Guided Genetic Programming algorithm which evolves, from a given grammar, an expression that maps to the scheduling of transmissions. We evaluate in our work the possibility of improving the results obtained by the state-of-the-art using a layered grammar approach. We show that starting with a small restricted grammar and introducing the full functionality after 10 generations outperforms the state-of-the-art, even when varying the algorithm used to generate the initial population and the maximum initial tree depth.

Keywords: Telecommunication, Heterogeneous Network, Scheduling, Grammar-Guided Genetic Programming, Multi-Level Grammar.

1 Introduction

We have seen in the last decade a proliferation in the use of mobile phones worldwide to reach 4.47 billion users in 2017 and this number is expected to exceed the 5 billion barrier by 2019 [1]. Companies attempt to attract new costumers through price cuts and the introduction of new technologies, like the soon-to-come 5G networks. Due to the heterogeneity and growing size of the networks, there is a large and increasing need to optimise their performance [2].

In traditional single cellular networks, Macro Cells (MCs) are employed to cover all User Equipments (UEs) such as phones, tablets, and any other device equipped with a broadband adapter. However, with the explosion of connected devices, MCs struggle to cope with the load. Therefore, they have to be supplemented with local and less powerful Small Cells (SCs), creating a two-tiered configuration called Heterogeneous Networks (HetNets). SCs are deployed in areas with traffic hot-spots to attract the near-by UEs, which offloads the MCs and

mitigates their performance deficit. Despite being beneficial from cost and performance points of view, SCs share the same bandwidth as the MCs, thus, making them more susceptible to interference. To mitigate possible interference, the 3^{rd} Generation Partnership Project (3GPP [3]) provisioned an enhanced Inter-Cell Interference Coordination (eICIC) mechanism i.e., Almost Blank Subframes (ABSs). The ABSs force MCs to mute periodically for a certain duration, allowing SCs to communicate with their UEs without interference from the near-by MCs.

Several challenges are induced by the configuration of these HetNets and require real-time optimisation. In our work, we particularly address the definition of the ABS and the scheduling of the UEs when communicating with their respective SC. The current state-of-the-art for optimising this problem in a real-time fashion (millisecond timescale) is using a Grammar-Guided Genetic Programming algorithm (G3P [4]). However, the designed algorithm starts with a randomly generated initial population and uses a unique and thorough grammar from the beginning to the end of the optimisation.

To improve the performance of evolutionary algorithms, some works in the literature use greedy techniques to generate good individuals as an initial population (e.g., [5,6]), while others promote an incremental introduction of the domain knowledge to the optimisation algorithm (for instance, McKay et al. [7] use a developmental strategy of the grammar in Genetic Programming, whereas NEAT [8] augments the typologies of neural networks.

Our work evaluates the advantage of using a succession of grammars during the evolution with incremental granularities instead of a single one. The idea is based on: (i) starting with a grammar that contains fewer terminals with the aim of guiding the optimisation towards individuals with 'ideal' forms, and (ii) introducing a larger and more thorough grammar after some generations with the aim of increasing the search space and thus improving the quality of the individuals further. We create a hybridisation of different G3P algorithms where the first ones are used to direct the search towards interesting individuals and the last one to probes the whole search space, similarly to [6,9,10]. Our work is organised around and aims at answering this main Research Question (mRQ): Is it good to use different grammar levels?

It has been shown by Nicolau [11] that the way the initial population is generated affects drastically the performance of grammar-based genetic programming algorithms. Therefore, we evaluate the way our approach is affected by the modification of two parameters related to the initial population, in two secondary Research Questions: (sRQ1) the algorithm used to generate the initial population, and (sRQ2) the maximum initial tree depth.

The rest of this paper is organised as follows: Section 2 defines the problem of scheduling in Heterogeneous Networks. Section 3 presents a short study of the works done on the problem. Section 4 describes the state-of-the-art algorithm G3P for the scheduling in HetNets, in addition to our multi-level grammar approach. In particular, we present the different grammars and the mapping of an expression to a schedule. Section 5 describes the dataset, the setup and the

significance test. Section 6 aims at answering the aforementioned research questions and shows results of the experiments. Section 7 concludes our study and proposes some future directions that we would like to explore.

2 Problem Definition

Let us consider a Heterogeneous Network $\mathcal{N}$ with a set of Macro Cells $\mathcal{M}$ and Small Cells $\mathcal{S}$. We also consider a set of User Equipements $\mathcal{U}$ and that every UE $u_i \in \mathcal{U}$ receives a signal σ_i^j from the cell $c_j \in \mathcal{M} \cup \mathcal{S}$.

2.1 Attaching UEs

UEs are known to attach greedily to the cell from which they receive the strongest signal. Since SCs have low power, the number of UEs that attach to them is limited. To cope with this issue, the 3GPP provisioned a bias mechanism i.e., Range Expansion Bias (REB) enabling SCs to attach a larger number of UEs beyond the area where their signal is higher than the near-by MCs. Therefore, the signal σ_i^j of every cell $c_j \in \mathcal{M} \cup \mathcal{S}$ to a UE $u_i \in \mathcal{U}$ is biased by an REB β_j, with $\beta_j = 0$ for every $c_j \in \mathcal{M}$. Every UE u_i is attached to a cell $c_j \in \mathcal{M} \cup \mathcal{S}$:

$$c_j = \arg\max_{k=1}^{|\mathcal{M}\cup\mathcal{S}|}(\sigma_i^k + \beta_k) \tag{1}$$

Definition 1. *Expanded Region E_j of an SC $c_j \in \mathcal{S}$ is the area where UEs would attach to c_j, but would not attach to it without the using the bias β_j. We say that a UE u_i is in the expanded region E_j an SC $c_j \in \mathcal{S}$ if and only if:*

$$c_j = \arg\max_{k=1}^{|\mathcal{M}\cup\mathcal{S}|}(\sigma_i^k + \beta_k) \quad \wedge \quad c_j \neq \arg\max_{k=1}^{|\mathcal{M}\cup\mathcal{S}|}(\sigma_i^k) \tag{2}$$

Figure 1 shows an example that summarises the aforementioned concepts.

2.2 Almost Blank Subframes

Using the same communication channel between MCs and SCs exasperates the interference at the expanded regions. The 3GPP framework defines a time domain (i.e., a frame $\mathcal{F}$) containing 40 subframes (SF) with a 1ms time interval for each subframe. The interference can be mitigated by muting the transmission of the MCs at some of the subframes using the ABS mechanism. Therefore, allowing near-by SCs to communicate with UEs in their expanded region with low interference. By muting the MCs during some SFs, UEs at the expanded regions experience a large reduction in interference. However, UEs attached to MCs cannot communicate with their respective cells during that time frame.

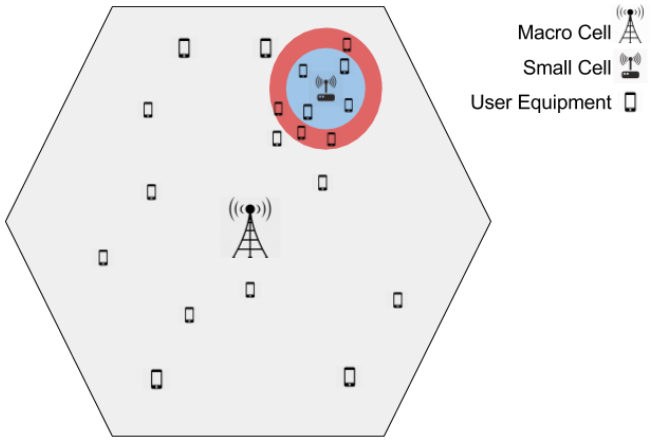

Fig. 1. Example of a Heterogeneous Network with one Macro Cell, one Small Cell, and 20 User Equipements. The grey hexagon corresponds to the area where UEs will attach to the MC in absence of any SC. The blue area is the region where the signal from the SC is stronger than the one coming from the MC. In red is the expanded region where UEs attach to the SC thanks to the bias.

2.3 Scheduling

The downlink rate R_i^f of a UE u_i quantifies the amount of data that can be transferred in the SF S_f. R_i^f is described by Shannon's formula [12] as depending on: (i) the bandwidth B, (ii) the number of UEs communicating at the same SF S_f, and (iii) the Signal to Interference and Noise Ratios (SINRs):

$$R_i^f = \frac{B}{N_f} \times log_2(1 + SINR_i^f) \tag{3}$$

UEs attached to MCs experience high SINR making their downlink always high. Therefore, scheduling UEs attached to MCs is trivial as they can all be allocated to all the SFs when the MCs are active (i.e., not muted). However, UEs attached to SCs experience a relatively low signal (SCs are low powered devices) and are subject to high interference from MCs (during their active SFs).

While the bandwidth is expensive and scarce and thus, hard to improve, both the $SINR$ and the number of communicating UEs N could be improved. The $SINR_i^f$ can be improved by muting MCs at the given SF S_f. However, exaggerating this process would lead to a substantial reduction in the overall downlink rate of UEs attached to MCs (which may be more numerous) as they would not receive any data in the mean time. Similarly, reducing the number of UEs communicating simultaneously and only communicating with fewer of them would improve the downlink for the active ones, but would mean that dismissed UEs will not be receiving any transmission. All these aspects make the management of transmissions not trivial, requiring an autonomic scheduling system that would specify: (i) SFs at which MCs are muted, and (ii) UEs communicating at any given SF.

2.4 Fitness Function

On average, every UE $u_i \in \mathcal{U}$ experiences an average downlink $\bar{R}_i$ over all the SFs of the same frame.

$$\bar{R}_i = \frac{1}{|\mathcal{F}|} \sum_{S_f \in \mathcal{F}} R_i^f \tag{4}$$

HetNets typically aim at optimising fairness of downlinks experienced by the different users in the network [13] including the state-of-the-art work [4] we are comparing to in this paper. This fairness is expressed as the sum of average downlink logs (i.e., $log(\bar{R}_i)$) of all the UEs:

$$Fairness = \sum_{u_i \in \mathcal{U}} log\left(\bar{R}_i\right) \tag{5}$$

Maximising the logs of average downlinks sets a high penalty when having UEs with low average downlinks, while at the same time does not provide a large reward when having UEs with excessively high average downlinks. In this work, we aim at maximising the fairness in Eq. 5 as the fitness function.

3 Previous Works

Most works in the literature to address the scheduling of transmission in HetNets put forward algorithms designed by expert agents. The most employed strategy is to partition the UEs that are attached to SCs into two different groups [14] based on the SFs they are scheduled to communicate at: (i) ABS-SFs; SFs in which MCs are muted or (ii) Non-ABS SFs; SFs in which MCs are active. Jiang and Lei [15] model the problem as a two-player bargaining game between ABS and Non-ABS SFs to attract the UEs to transmit within their time intervals. Lopez et al. [16] aim at balancing the downlinks for the UEs in both groups to equalise each other.

Autonomic solutions for the scheduling problem in HetNets are only proposed in the recent years. Lynch et al. [4] use a Grammar-Guided Genetic Programming algorithm [17] to evolve the schedules in a reinforcement learning fashion. Their algorithm has been proven to outperform the state-of-the-art systems designed by the experts across multiple metrics. The authors also showed by running a genetic algorithm for a longer period (not real-time) a large potential for performance improvement, and this is one of the major motivations for our work. The authors use a single full grammar during the entire evolution process. In our work, we propose feeding the algorithm with a smaller and more compact grammar, before extending it during the evolution (after some generations).

There has been much research into grammars. Many different approaches have been proposed and investigated from probabilistic grammars, where each production updates the probability of the production being allowed to happen again, to developmental evaluation [7] which evolves the grammar during the evolution. A comprehensive survey of these methods is presented by Hemberg [18].

With all these approaches the idea of layered learning is key. The goal is to learn during evolution and then use that knowledge to bootstrap to the next solution. In a similar vein, the approach proposed in this paper looks to establish a strong corpus of individuals that are then allowed to explore the much wider search space of the unrestricted grammar.

4 Multi-Level Grammar-Guided Genetic Programming

In this section, we describe both the state-of-the-art algorithm for scheduling in HetNets (i.e., G3P) and our proposed approach (i.e., multi-level grammar).

4.1 State-of-the-Art: Grammar-Guided Genetic Programming

The state-of-the-art for the scheduling in HetNets is a Grammar-Guided Genetic Programming [4] algorithm which uses a unique grammar in a Backus-Naur Form (BNF) to incorporate domain knowledge:

```
<expr> ::= <reg> | <reg> | <reg> | <Terminal>
<reg> ::= <expr><op><expr> | <expr><op><expr> | <expr><op><expr> | <expr><op><expr> |
          <non-linear>(<expr>) | <non-linear>(<expr>)
<op> ::= + | - | * | / (protected)
<non-linear> ::= sin | log (protected) | sqrt (protected) | step
<Terminal> ::= <sign><const> | <statistic>
<sign> ::= - | +
<const> ::= 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0
<statistic> ::= downlink | num_variable | num_att | airtime | congestion |
                avg_downlink_frame | max_downlink_frame | min_downlink_frame |
                avg_downlink_SF | max_downlink_SF | min_downlink_SF |
                avg_downlink_cell | max_downlink_cell | min_downlink_cell
```

While most of the rules in this grammar are common to the GP world and easy to understand, <statistic> contains terminals that are from the network domain and we refer the reader to [4] for their formal definition.

The state-of-the-art algorithm is G3P: an adaptation of a grammar-based form of GP [19] as implemented in the PonyGE 2 framework [20]. G3P is used to evolve an expression that maps the SINR related statistics and attachment information to a binary decision for each UE per SF: whether to schedule the UE to communicate at the given SF or not. The authors use Algorithm 1 (please refer to [4] for a more detailed version) to do this mapping, before evaluating the fitness function with the resulting schedule.

4.2 Our Approach: Multi-Level Grammar

In addition to the full and more thorough grammar (i.e., F) defined by the state-of-the-art, we define two other grammars by only updating the list of available terminals. We have created two incremental grammars: (i) S: small, and (ii) M: medium, such that S is included in M and M is included in F.

The small grammar is defined by modifying <const> and <statistic>. The number of terminals is reduced to the strict minimum by only keeping a small

```
input  : E: Expression
output: M: Schedule Matrix
for c_j ∈ S do
    M[j] ← zeros(|F| × |U|) // define a transmission schedule matrix
    for S_f ∈ F do
        for u_i ∈ U do
            interest ← evaluate(E, M, i, f) // evaluate expression for u_i
                in f with current Schedule
            if interest > 0 and SINR_i^f ≥ 1 then
                M_j[j][i][f] ← 1 // set as 'scheduled'
            end
        end
    end
end
return N;
```

Algorithm 1: Mapping of an expression to a transmission schedule.

subset of constants and what seems to be the most important statistics. The downlink is what we would like to optimise. Whereas maximising the value of min_downlink_frame would improve the smallest downlinks. Therefore, improving it would have a better impact on the fitness function. We set in S:

```
<const> ::= 0.0 | 0.5 | 1.0
<statistic> ::= downlink | min_downlink_frame
```

The medium grammar is also defined by modifying <const> and <statistic>. We add 6 terminals to the medium grammar: 4 constants (2 signs × 2 constants) and 2 statistics (i.e., max_downlink_frame and min_downlink_cell) that are also related to the downlink, in addition to the terminals from the grammar S. We set <const> and <statistic> in M as follows:

```
<const> ::= 0.0 | 0.3 | 0.5 | 0.8 | 1.0
<statistic> ::= downlink | min_downlink_frame | max_downlink_frame | min_downlink_cell
```

After defining these grammars (i.e., S, M and F), we adapted the state-of-the-art algorithm G3P to take one grammar at the start of the experiment and dynamically modify the grammar to a more complex one (e.g., from S to M, M to F, or S to F). All individuals obtained using a given grammar are seeded as an initial population [21] to G3P using the following grammar. We do not require any modification in the representation of the individuals when updating the grammar as they are represented in a tree form and the grammars are included within each other. This means that an individual has both the same representation and the same interpretation (in terms of schedule), before and after changing the grammar.

Although modifying the values of some parameters would have probably been ideal when introducing a new grammar (e.g., increasing the mutation rate for few generations facilitates the introduction of new terminals), we chose to not modify any parameter. We make this choice in order to mitigate any implication

from changing the values of these parameters and only leave one varying element at the time (i.e., the grammar).

5 Experiment Design

We describe in this section the dataset and the setup used in our experiment, in addition to the test used to assess the significance of our results.

5.1 Dataset

We simulate in our work three HetNets following the same process as described in [4] and all of them serving the same geographical area that encompasses 3.61 km^2 of Dublin city centre. All the HetNets contain 21 MCs spread in a hexagonal pattern. However, they differ in the number of SCs they contain. The first HetNet is the least dense with 21 SCs (1 SC per MC on average). The second HetNet is denser than the first one with 63 SCs (3 SCs per MC on average). The third and last HetNet is the densest among them with 105 SCs (5 SCs per MC on average). Additionally, we consider that a total of 1250 UEs are in the considered geographical area and are attached to one of the MCs or SCs.

5.2 Setup

We use the state-of-the-art algorithm G3P provided by the authors [4]. To validate our approach, we design different grammar configurations (see Section 6). We compare the best fitness function obtained when using each of the configurations in G3P instead of the full grammar. We set the population size to 100 and allow the algorithm to run for 100 generations. Furthermore, we use the Ramped Half-Half (RHH [22]) algorithm to generate the initial population with a maximum tree depth of 20. We use the sub-tree crossover with a probability 0.5, and undergo a sub-tree mutation to 60% of the population, while point mutating the remaining 40%. We set all the other parameters as described in [4]. Moreover, we repeat every experiment 30 times to minimise the effect of randomness.

5.3 Significance

In order to validate the significance of our comparisons, we perform a statistical test using a non-parametric test: the two-tailed Mann-Whitney U test (MWU). In every experiment, MWU takes in the different performance values (best fitness function values) obtained by two algorithms from each run (i.e., 30). MWU returns the p-value that one of the algorithms obtains different values than the other. We consider tests significant when the p-value is below 0.05.

6 Evaluation

We aim in this section to answer the research questions that were formalised in Section 1 experimentally.

6.1 mRQ: Is it good to use different grammar levels?

In order to show the relevance of combining different grammars, we compare 7 grammar configurations on the three instances (21 SCs, 63 SCs and 105 SCs). We designed 6 different grammar configurations in addition to the default scenario F (one full grammar from beginning to end):

- S5M10F: start with S and introduce M and F at generations 5 and 10.
- S10M20F: start with S and introduce M and F at generations 10 and 20.
- S5M20F: start with S and introduce M and F at generations 5 and 20.
- S1F: start with S and introduce F at generation 1 (after generation 0).
- S5F: start with S and introduce F at generation 5.
- S10F: start with S and introduce F at generation 10.

We set parameters of G3P to the same values over all the grammar configurations as described in Section 5.

Figure 2 shows the evolution per generation of the best fitness on each instance, obtained by G3P when using the different grammar configurations (results are averaged over 30 runs).

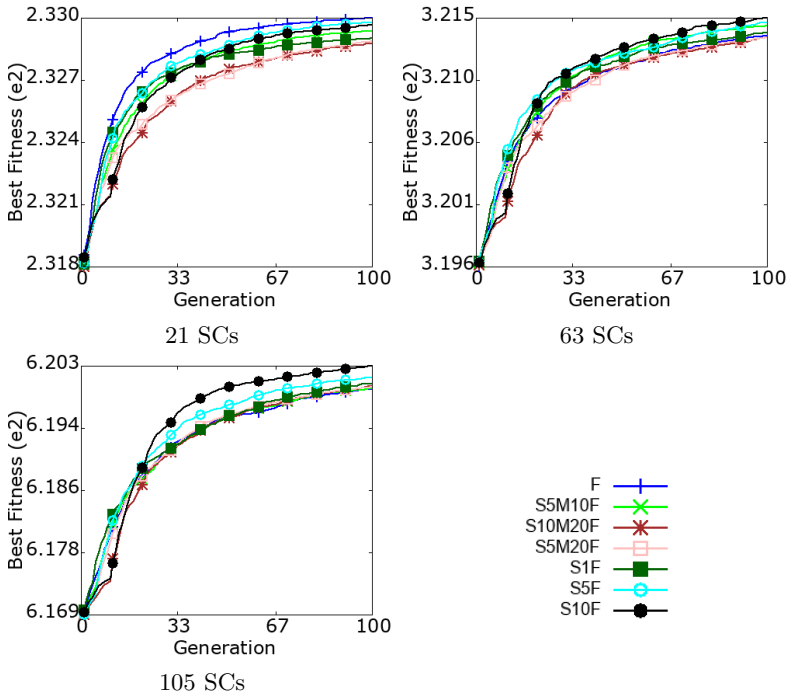

Fig. 2. Average over 30 runs of the evolution of the best fitness obtained by G3P on the different instances using various grammar configurations.

We notice from Figure 2 that G3P improves the best fitness function for all instances (constantly improves the baseline i.e., the smallest recorded values:

231.764, 319.588 and 616.874 for 21 SCs, 63 SCs and 105 SCs respectively) regardless of the grammar configuration it is used with. We also notice that the number of generations (i.e., 100) is not enough to achieve a full convergence of the algorithms and increasing this parameter would allow achieving a better performance –but would increase the execution time though.

We see that using the full grammar only (i.e., F) achieves the best results on the 21 SCs instance (outperforming the second best grammar S5F with 1.69% on average). However, its performance worsens significantly on the two other instances (i.e., 63 SCs and 105 SCs) where S10F achieves the best results (S10F achieves 7.54% and 9.95% better results than F on average on 63 SCs and 105 SCs respectively). S5F also achieves similar results as S10F. Although S5F does not reach the same quality of results as S10F on 63 SCs and 105 SCs, it slightly outperforms it on 21 SCs. Despite S1F being based on the same principle as S5F and S10F (only using the small and the full grammar), it does not achieve good results. This is mostly due to the fact that using the small grammar for only one generation only affects the individuals in the initial population. S1F generates individuals with phenotypes composed of a smaller set of terminals and does not aim at converging towards 'ideal' individuals. This acts almost as a handicap for the evolution as it does not provide either the greediness to converge faster or the variety to explore the search space. Whereas, S5F and S10F optimise the initial population further using the same small set of terminals, thus exploiting this small number of terminals for a better convergence.

Surprisingly, algorithms which use a succession of three grammar levels do not achieve as good results as those only using two grammars, with the exception of S5M10F. This is even more surprising as S10M20F achieves the worst performance in almost all instances. This goes against the intuition that if we have a grammar configuration (in this case S10M20F) similar to another one which achieves good results (e.g., S10F), the former is likely to achieve a good performance as well. If we look closely at the improvement curves of S10F and S10M20F at about 10 generations, they all seem to converge (or at least improve slowly). The introduction of the new grammar (i.e., a full grammar in the case of S10F) enables it to introduce new terminals to the evolution and improve drastically its performance. Whereas the introduction of the medium grammar allows S10M20F a much more limited improvement. In case of S10M20F, we notice a second convergence/stagnation around 20 generations before the full grammar gets introduced. However, given the limited number of generations, G3P with S10M20F could not reach the improvement of the other ones. We believe, however, that given a larger number of generations, this configuration could improve its performance.

Table 1 shows the mean and standard deviation over 30 runs of the best fitness function on the different instances, achieved by G3P when using the aforementioned grammar configurations. It also includes the p-value when comparing every approach against G3P with the grammar configuration F.

Table 1 confirms what has been noticed in Figure 2 and shows that F achieves the best results on 21 SCs on average, whereas S10F achieves the best results on

Instance	Function	F	S5M10F	S10M20F	S5M20F	S1F	S5F	S10F
21 SCs	Mean	**233.025**	232.963	232.903	232.915	232.927	233.004	232.991
	Stdev	0.038	0.088	0.082	0.090	0.072	0.065	0.128
	p-value	-	**1.89E-04**	**3.56E-09**	**4.92E-08**	**5.78E-08**	1.56E-01	3.81E-01
63 SCs	Mean	321.341	321.415	321.330	321.328	321.367	321.439	**321.473**
	Stdev	0.172	0.132	0.162	0.142	0.135	0.224	0.188
	p-value	-	**3.51E-02**	3.42E-01	2.55E-01	3.10E-01	**4.52E-04**	**7.59E-04**
105 SCs	Mean	619.950	619.953	619.990	619.972	620.022	620.104	**620.256**
	Stdev	0.261	0.273	0.225	0.176	0.269	0.321	0.299
	p-value	-	4.09E-01	2.46E-01	3.81E-01	1.03E-01	**3.19E-03**	**6.68E-06**

Table 1. Mean and standard deviation over 30 runs of the best fitness obtained by G3P when using different grammars. In addition, we include the p-value (using MWU) in comparison to the results obtained against G3P with grammar F. Note that we put '-' when computing the p-value for F against F as it is always 0.5 and thus not worth including in the results. We put in bold the best mean and significant p-values.

average on both 63 SCs and 105 SCs. It also shows that the standard deviation is rather large with regards to the difference in means. However, the p-value confirms that results obtained with S10F and S5F are significantly better than those obtained with F on both 63 SCs and 105 SCs. Whereas, the results obtained with F on 21 SCs are not significantly better than those of S10F and S5F.

The answer to mRQ is: Yes. It is good to use different grammar levels in most cases. More particularly in our case, starting with the small grammar and introducing the full one after 10 generations is the best grammar configuration.

6.2 sRQ1: How are the results affected by the algorithm used to generate the initial population?

We have shown from the previous research question that using S10F as a grammar configuration for G3P achieves significantly better results than using F in most cases. However, we would like to check whether these results are dependent on the way we generate the initial population. Therefore, we compare both grammar configurations F and S10F by varying the algorithms used to generate the initial population and fixing the other parameters to the same values. We use four different initialisation algorithms: (i) RHH: Ramped Half-Half [22], (ii) PIG: Position-Independent Grow [23], (iii) UT: Uniform Tree [23], and (iv) UG: Uniform Genome [23].

Figure 3 shows the evolution per generation of the best fitness on each instance, obtained by G3P when using either F or S10F with different initialisation algorithms (results are averaged over 30 runs).

We see from Figure 3 that G3P successfully improves the fitness function over the 100 generations regardless of the grammar configuration and the algorithm used for the initialisation. We also see that the results do not fully converge within the 100 generations and that increasing this parameter is likely to improve the results.

In terms of performance, we clearly see that G3P achieves the best results when using RHH to generate the initial population on most instances, except

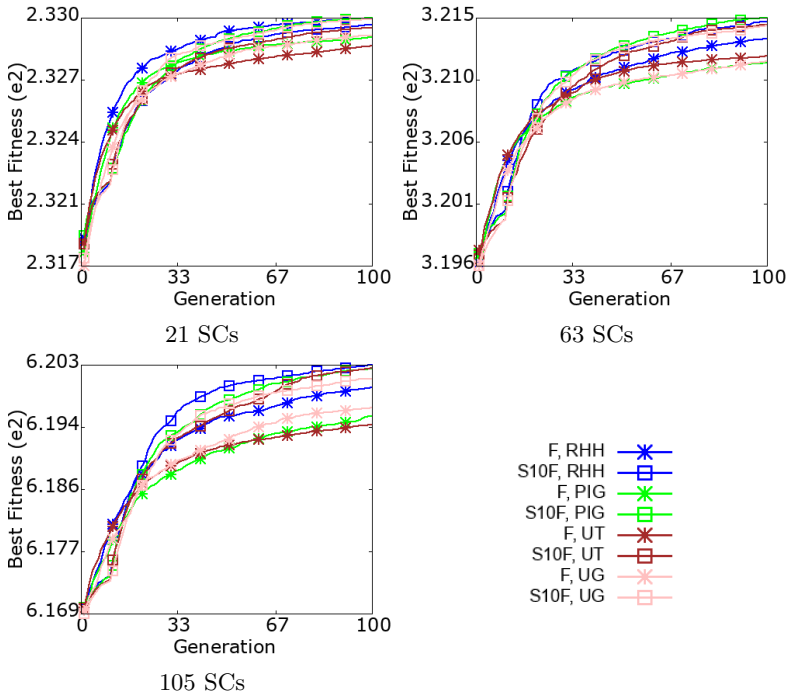

Fig. 3. Average over 30 runs of the evolution of the best fitness obtained by G3P on the different instances using either the full grammar (i.e., F) or the two-level grammar (i.e., S10F), when generating the initial populations with various initialisation algorithms (i.e., RHH, PIG, UT and UG).

63 SCs. This validates the default setting chosen for the state-of-the-art algorithm [4]. We also see that with the exception of RHH on 21 SCs, using S10F allows getting better results than F. This is an important indicator that using the two-level grammar is better than the single grammar regardless of the algorithm used to generate the initial population. We even notice that S10F outperforms F on all instances and achieves better results when using initialisation algorithms different from RHH (S10F achieves 4.93%, 17.95%, 17.64% and 12.73% better results on average than F when using respectively RHH, PIG, UT and UG).

Table 2 shows the mean and standard deviation over 30 runs of the best fitness function on the different instances achieved by G3P when using either F or S10F as a grammar configuration, and when varying the algorithm used to generate the initial population. It also includes the p-value (using MWU) between the results with F and S10F in each scenario.

Table 2 confirms that using S10F always leads to better results than F regardless of the algorithm used to generate the initial population (except with RHH on 21 SCs). We also notice a relatively large standard deviation in comparison to the difference in mean values. However, the standard deviation is similar to both grammar configurations. Despite the large standard deviations, using

Instance	Function	RHH		PIG		UT		UG	
		F	S10F	F	S10F	F	S10F	F	S10F
21 SCs	Mean	**233.025**	232.991	232.926	**233.027**	232.882	**232.976**	232.936	**233.019**
	Stdev	0.038	0.128	0.057	0.042	0.064	0.073	0.081	0.053
	p-value	3.81E-01		**2.16E-08**		**5.09E-06**		**3.83E-05**	
63 SCs	Mean	321.341	**321.473**	321.161	**321.501**	321.210	**321.450**	321.173	**321.442**
	Stdev	0.172	0.188	0.113	0.096	0.102	0.093	0.109	0.118
	p-value	**7.59E-04**		**1.84E-11**		**2.10E-10**		**7.05E-10**	
105 SCs	Mean	619.950	**620.256**	619.567	**620.201**	619.448	**620.210**	619.676	**620.071**
	Stdev	0.261	0.299	0.277	0.256	0.285	0.194	0.269	0.307
	p-value	**6.68E-06**		**4.24E-09**		**3.35E-11**		**2.99E-05**	

Table 2. Mean and standard deviation over 30 runs of the best fitness obtained by G3P using either F or S10F as grammar configuration, when varying the algorithm used to generate the initial population. In addition, we include the p-value (using MWU) to test the significance of the results. We put in bold the best mean between F and S10 F and significant p-values.

S10F allows achieving significantly better results with respect to the MWU test. Furthermore, we see that the unique case where F achieves better mean results than S10F is not significant.

The answer to sRQ1 is: The two-level grammar S10F allows achieving even better performance in comparison to the full grammar when used with different initialisation algorithms.

6.3 sRQ2: How are the results affected by the maximum initial tree depth used to generate the initial population?

We have confirmed in the previous research questions that we achieve a better performance when using the two-level grammar S10F against when using only the full one (i.e., F). We have also confirmed that the results are not biased by the choice of the initialisation algorithm, as they are better with all the of them (except in case of RHH with 21 SCs).

In this part, we attempt to confirm whether the quality of results obtained using the two-level grammar is not negatively impacted by the maximum depth of the initial trees. We, therefore, run G3P using both F and S10F, with various maximum initial tree depths (i.e., 5, 10, 20 and 30), while setting the other parameters to their default values (more particularly, the algorithm used to generate the initial population is set to RHH).

Figure 4 shows the evolution per generation of the best fitness on each instance, obtained by G3P when using either F or S10F with different maximum depths for the initial trees (averaged over 30 runs).

We see from Figure 4 that G3P improves the fitness function with all the maximum initial tree depths without fully converging within the 100 generations. We see that using the depth 20 achieves the best results only on the instance 21 SCs, whereas the maximum depth 10 achieves the best results on the other ones. Similarly to what has been noticed when varying the initialisation algorithm, S10F outperforms F in all the cases (instances $\times$ maximum

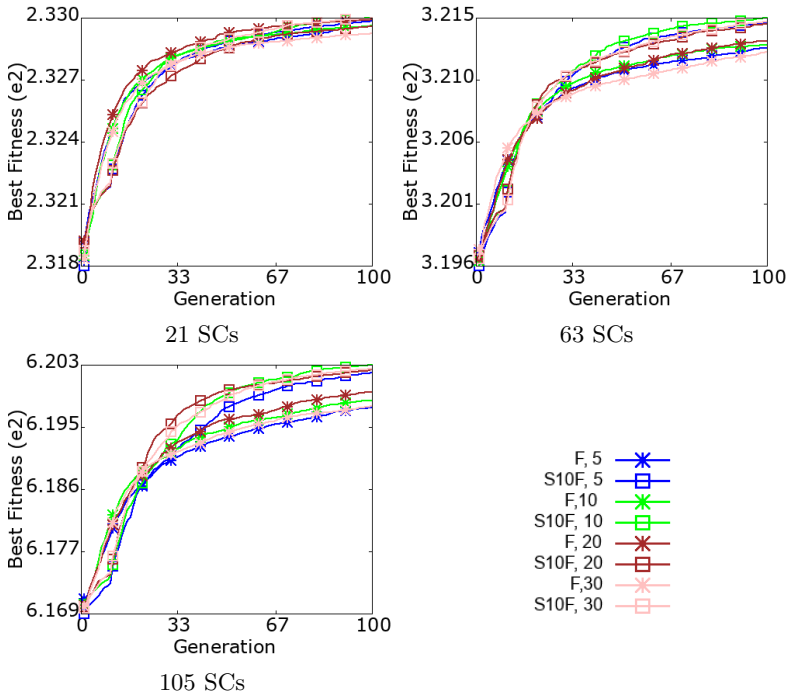

Fig. 4. Average over 30 runs of the evolution of the best fitness obtained by G3P on the different instances using either the full grammar (i.e., F) or the two-level grammar (i.e., S10F), when generating the initial populations with various maximum initial tree depths (i.e., 5, 10, 20 and 30).

initial tree depths) except in 21 SCs with a depth of 20. This clearly shows that using the two-level grammar S10F is better than the full one and that it is not biased by the maximum initial tree depth. In addition to the fact that the value of maximum initial tree depth affects the final results, it also seems to affect the difference between the two grammars (i.e., S10F outperforms F more when using a maximum depth of 30 with 12.93% improvement than when using a maximum depth of 20 with 4.93% improvement on average).

Table 3 shows the mean and standard deviation over 30 runs of the best fitness function by G3P with either F or S10F as a grammar configuration while varying the maximum initial tree depth. It also includes the p-value (using MWU) between results with F and S10F in each scenario. As with Table 2, Table 3 confirms that using S10F always leads to significantly (i.e., based on MWU) better results than F regardless of the maximum initial tree depth (except with depth 20 on 21 SCs).

The answer to sRQ2 is: The two-level grammar S10F allows achieving even better performance in comparison to the full grammar when used with different maximum initial tree depths.

Instance	Function	5		10		20		30	
		F	S10F	F	S10F	F	S10F	F	S10F
21 SCs	Mean	232.991	**233.018**	232.993	**233.035**	**233.025**	232.991	232.954	**233.033**
	Stdev	0.044	0.060	0.050	0.035	0.038	0.128	0.055	0.051
	p-value	**3.62E-02**		**2.54E-03**		3.81E-01		**7.15E-06**	
63 SCs	Mean	321.288	**321.485**	321.310	**321.521**	321.341	**321.473**	321.255	**321.491**
	Stdev	0.085	0.100	0.080	0.086	0.172	0.188	0.067	0.090
	p-value	**1.75E-09**		**4.88E-10**		**7.59E-04**		**2.49E-11**	
105 SCs	Mean	619.731	**620.213**	619.833	**620.324**	619.950	**620.256**	619.750	**620.266**
	Stdev	0.194	0.221	0.176	0.141	0.261	0.299	0.214	0.208
	p-value	**4.24E-09**		**6.64E-11**		**6.68E-06**		**9.28E-10**	

Table 3. Mean and standard deviation over 30 runs of the best fitness obtained by G3P using either F or S10F as grammar configuration, when generating the initial populations with various maximum initial tree depths (i.e., 5, 10, 20 and 30). We also include the p-value (using MWU) to test the significance of the results. We put in bold the best mean between F and S10F and significant p-values.

7 Conclusion

We studied the use of different levels of grammars as a mean to improve the quality of the schedules in HetNets obtained by the G3P algorithm. Our approach consists of starting the optimisation with a short grammar which contains only the most important terminals in order to direct the search towards 'ideal' individuals. Then, to introduce a more thorough grammar during the evolution.

We showed that starting with the small grammar and introducing the full one after 10 generations, allows us to outperform the standard configuration which only uses one full grammar with up to 10% on average. We also showed that our approach is better in most cases regardless of the initialisation algorithm and maximum initial tree depth used to generate the initial populations.

In the future, we would like to analyse the sensitivity of the approach towards the number and the quality of the terminals in the small grammar. We also would like to investigate whether adding a local search to the two-level grammar, creating a three-step method [6], would be beneficial. Furthermore, we would like to study the way the approach is affected when the grammar is modified at positions other than terminals. Moreover, the quality of results obtained on the scheduling in HetNets motivates us to study the performance of our two-level grammar approach on problems from other domains.

Acknowledgement

This research is based upon works supported by the Science Foundation Ireland under Grant No. 13/IA/1850.

References

1. Number of mobile phone users worldwide from 2013 to 2019 (in billions). www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide

2. Andrews, J.G., Buzzi, S., Choi, W., Hanly, S.V., Lozano, A., Soong, A.C., Zhang, J.C.: What will 5g be? IEEE Journal on Selected Areas in Communications 32(6), 1065–1082 (2014)
3. 3gpp: The 3rd generation partnership project. www.3gpp.org
4. Lynch, D., Fenton, M., Kucera, S., Claussen, H., O'Neill, M.: Scheduling in heterogeneous networks using grammar-based genetic programming. In: EuroGP. pp. 83–98 (2016)
5. Saber, T., Marques-Silva, J., Thorburn, J., Ventresque, A.: Exact and hybrid solutions for the multi-objective vm reassignment problem. International Journal on Artificial Intelligence Tools 26, 1760004 (2017)
6. Saber, T., Ventresque, A., Gandibleux, X., Murphy, L.: Genepi: A multi-objective machine reassignment algorithm for data centres. In: HM. pp. 115–129 (2014)
7. McKay, R.I., Hoang, T.H., Essam, D.L., Nguyen, X.H.: Developmental evaluation in genetic programming: the preliminary results. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP. pp. 280–289 (2006)
8. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
9. Saber, T., Ventresque, A., Brandic, I., Thorburn, J., Murphy, L.: Towards a multi-objective vm reassignment for large decentralised data centres. In: UCC. pp. 65–74 (2015)
10. Saber, T., Thorburn, J., Murphy, L., Ventresque, A.: Vm reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge. Future Generation Computer Systems 79, 751–764 (2018)
11. Nicolau, M.: Understanding grammatical evolution: initialisation. Genetic Programming and Evolvable Machines 18, 467–507 (2017)
12. Shannon, C.E.: Communication in the presence of noise. IRE 37(1), 10–21 (1949)
13. Weber, A., Stanze, O.: Scheduling strategies for hetnets using eicic. In: ICC. pp. 6787–6791 (2012)
14. Fagan, D., Fenton, M., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: Deep learning through evolution: A hybrid approach to scheduling in a dynamic environment. In: IJCNN. pp. 775–782 (2017)
15. Jiang, L., Lei, M.: Resource allocation for eicic scheme in heterogeneous networks. In: PIMRC. pp. 448–453 (2012)
16. Lopez-Perez, D., Claussen, H.: Duty cycles and load balancing in hetnets with eicic almost blank subframes. In: PIMRC. pp. 173–178 (2013)
17. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in grammatical evolution for dynamic environments, vol. 194. Springer (2009)
18. Hemberg, E.A.P.: An Exploration of Grammars in Grammatical Evolution. Ph.D. thesis, University College Dublin (2010)
19. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines 11, 365–396 (2010)
20. Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E., O'Neill, M.: Ponyge2: Grammatical evolution in python. In: GECCO. pp. 1194–1201 (2017)
21. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Is seeding a good strategy in multi-objective feature selection when feature models evolve? Information and Software Technology (2017)
22. Ryan, C., Azad, R.M.A.: Sensible initialisation in grammatical evolution. In: GECCO. pp. 142–145 (2003)
23. Fagan, D., Fenton, M., O'Neill, M.: Exploring position independent initialisation in grammatical evolution. In: CEC. pp. 5060–5067 (2016)