Genetic Programming and Evolvable Machines (2019) 20:245–283
https://doi.org/10.1007/s10710-019-09346-4

A multi-level grammar approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks

Takfarinas Saber[1] · David Fagan[2] · David Lynch[2] · Stepan Kucera[3] · Holger Claussen[3] · Michael O'Neill[2]

Received: 29 June 2018 / Revised: 16 February 2019 / Published online: 1 March 2019

Abstract

The scale at which the human race consumes data has increased exponentially in recent years. One key part in this increase has been the usage of smart phones and connected devices by the populous. Multi-level heterogeneous networks are the driving force behind this mobile revolution, but these are constrained with limited bandwidth and over-subscription. Scheduling users on these networks has become a growing issue. In recent years grammar-guided genetic programming (G3P) has shown its capability to evolve beyond human-competitive network schedulers. Despite the performance of the G3P schedulers, a large margin of improvement is demonstrated to still exist. In the pursuit of this goal we recently proposed a multi-level grammar approach to generating schedulers. The complexity of the grammar was increased at various stages during evolution, allowing for individuals to add more complex functions through variation operations. The goal is to evolve good quality solutions before allowing the population to specialise more as the grammar functionality increased in a layered learning way. In this paper the results of this initial study are replicated, and confirmed, and it is seen that this approach improves the quality of the evolved schedulers. However, despite the gain in performance, we notice that the proposed approach comes with an acute sensitivity to the generation at which the grammar complexity is increased. Therefore, we put forward a novel seeding strategy and show that the seeding strategy mitigates the shortcomings of the original approach. The use of the seeding strategy outperforms the original approach in all the studied cases, and thus yields a better overall performance than the state-of-the-art G3P generated schedulers.

Keywords Telecommunication · Heterogeneous network · Scheduling · Grammar-guided genetic programming · Multi-level grammar · Seeding

✉ Takfarinas Saber
takfarinas.saber@ucd.ie

Extended author information available on the last page of the article

1 Introduction

Technological advancements such as The Cloud and smartphones have shifted the amount of interactions people have with their digital world [1, 2]: people upload personal data to 'The Cloud', stream music and movies, use on-line banking, etc. This shift permitted the download and streaming of multimedia content to outrank Voice over IP applications and services in terms of data exchange [3].

The number of smartphone users is in a continuous progression and is expected to reach a 5 billion high by 2019 [4]. While the increasing amount of data and processing power have been dealt with through the adoption of more suitable Big Data techniques [5], the volume of interactions with the digital world and the criticality of the on-line services continue to stress the wireless communication network infrastructure. This has driven network operators away from operational cost reduction to network capacity improvement [6]. Network operators value any gain in performance as it allows them to deliver and improve on their Quality of Service levels. Improved Quality of Service often attracts new users, thus keeps the company relevant in what is a highly competitive trillions of dollars industry [7].

Network operators also upgrade their infrastructure to more performing technologies [2]. While investments in new wireless technologies allow significant performance improvements, they also lead to the heterogeneity of their infrastructure. Traditional cellular networks solely use Macro Cells (MCs) to provide User Equipments (UEs, e.g., a smartphone, a tablet or a device that has a broadband adaptor) with a transmission capability. However, to cope with the increasing number of devices, network operators introduce Small Cells (SCs, small and low-powered cells) to their infrastructure. SCs are often deployed in areas with recurrent traffic hot-spots (e.g., town squares and parks) to attract clustered UEs in their surrounding, thus mitigating the performance deficit of MCs.

However, the introduction of SCs leads to a two-tiered infrastructure known as heterogeneous network (HetNet) which requires optimisation [8]. Besides, SCs share the same bandwidth as MCs, which makes them more prone to interfere with each other. In order to mitigate this interference, the 3rd Generation Partnership Project [9] has provisioned an enhanced Inter-Cell Interference Coordination mechanism called Almost Blank Sub-frames (ABSs). During these Almost Blank Sub-frames, MCs are muted for a defined duration, whereas SCs are allowed to communicate with their attached UEs with no interference from MCs. Moreover, the rapid and constant evolution of the demand and location of the users render any non-real-time algorithm not applicable in production environments.

Our work addresses the particular problem of defining the Almost Blank Sub-frames and scheduling the communication of UEs with their attached SCs. The state-of-the-art for optimising the scheduling of transmissions in HetNets in a real-time fashion is a Grammar-Guided Genetic Programming algorithm (G3P [10]). G3P evolves individuals (i.e., expressions) that are then mapped to transmission schedulers. The mapped schedulers are then deployed in a HetNet to generate transmission schedules (a matrix of Booleans which represents whether a UE i will communicate with its attached cell j at a given time) in a millisecond timescale. Globally, G3P could be seen as a method for building a millisecond transmission scheduler. While the authors achieve an important improvement with respect to the quality of the former expert-designed state-of-the-art scheduler (with the quality of schedulers assessed based on the fairness of their generated schedules as shown in Sect. 2), they also showed a large potential for further improvements as they have shown the existence of better transmission schedules using a genetic algorithm with no execution time restriction. This large potential is the main motivator to further their work and the premise for ours.

We have proposed in a recent work [11] to use a succession of grammars with incremental granularities during the evolution instead of a single grammar as a mean to improve the quality of schedules that are generated by the schedulers that are mapped from the G3P's evolved expressions, thus improving the overall performance of G3P. The multi-level grammar approach is based on: (1) starting with a grammar that contains fewer terminals with the aim of guiding the optimisation towards individuals with 'ideal' forms, and (2) 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 have adapted the G3P algorithm from Lynch et al. [10] to start with a small grammar, before introducing a more thorough grammar. During the grammar update, G3P 'ports' the entire population that resulted from the first grammar level as an initial population to the next grammar level.

We confirm in the current work that G3P with a multi-level grammar approach achieves better results than with a single grammar. However, we also show that the multi-level grammar approach using the population porting strategy is highly sensitive to the phase (i.e., generation) at which the large grammars are introduced. Therefore, we propose an alternative that uses the multi-level grammar approach with the seeding of the best individual at the end of every level. G3P with the multi-level grammar approach using the seeding strategy is demonstrated (1) to achieve better results, as well as (2) to be more resilient towards the introduction phase than using the previous population porting strategy.

The rest of this manuscript is organised as follows: Sect. 2 formally defines the scheduling in the heterogeneous networks problem. Section 3 presents the work related to both the scheduling in heterogeneous networks and the multi-level grammar-based genetic programming. Section 4 presents the multi-level grammar approach, while Sect. 5 describes the experimental design and the testbed used throughout our evaluation. Section 6 shows the improvement that could be achieved using the multi-level grammar instead of the unique grammar and identifies the shortcomings of using the population porting strategy during the grammar update. Section 7 introduces the seeding strategy and shows that it successfully covers the shortcomings observed using the population porting strategy. Section 8 concludes this work.

2 Problem definition

Let us consider a heterogeneous network $\mathcal{N}$ which is composed of a set of cells $\mathcal{C}$. The set of cells $\mathcal{C}$ contains both a set of macro cells $\mathcal{M}$ and a set of small cells $\mathcal{S}$. Furthermore, we consider a set of user equipements $\mathcal{U}$ with each UE $u_i \in \mathcal{U}$ receiving a wireless signal strength of σ_i^j from every cell $c_j \in \mathcal{C}$.

2.1 Heterogeneous networks

UEs are usually greedy as they customarily attach to the cell that has the strongest wireless signal at the UEs' geographical location. Given that SCs are low powered devices, only a limited number of UEs attach to SCs based on signal strength. Therefore, to increase the range at which UEs attach to SCs, the 3rd Generation Partnership Project provisioned a bias mechanism i.e., Range Expansion Bias (REB). This mechanism enables SCs to attach UEs even beyond the area where their signal is stronger than near-by MCs. This is achieved through the biasing of the signal σ_i^j of every cell $c_j \in \mathcal{C}$ to a UE $u_i \in \mathcal{U}$ by a value β_j, with $\beta_j = 0$ for every $c_j \in \mathcal{M}$. Hence, every UE u_i attaches to a cell $c_j \in \mathcal{C}$ such that:

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

Every cell $c_j \in \mathcal{C}$ attach a set of UEs $\mathcal{A}_j$:

$$\mathcal{A}_j = \left\{ \arg\max_{k=1}^{|\mathcal{C}|}(\sigma_i^k + \beta_k) = c_j \right\} \tag{2}$$

Definition 1 Expanded Region E_j of a SC $c_j \in \mathcal{S}$ is the area in which UEs would attach to c_j when using the bias β_j, but the same UE would not attach to c_j with no use of bias. Therefore, a UE u_i belongs to the expanded region E_j of a SC $c_j \in \mathcal{S}$ if and only if:

$$c_j = \arg\max_{k=1}^{|\mathcal{C}|}(\sigma_i^k + \beta_k) \quad \wedge \quad c_j \neq \arg\max_{k=1}^{|\mathcal{C}|}(\sigma_i^k) \tag{3}$$

MCs and SCs use the same communication channel. This shared communication channel exasperates the interference at the edge of the expanded regions. Therefore, the 3rd Generation Partnership Project puts in place a time domain (i.e., a frame $\mathcal{F}$). Each of the frames is composed of 40 sub-frames (SF) of 1ms time interval each. Using this time domain and the Almost Blank Sub-frames mechanism, transmission of MCs could be muted at some of the sub-frames. Muting MCs would mitigate the interference and allow near-by SCs to communicate with UEs at their expanded region with low disturbance. However, while UEs at the expanded regions experience a large decrease in interference when muting MCs for some SFs, UEs that are

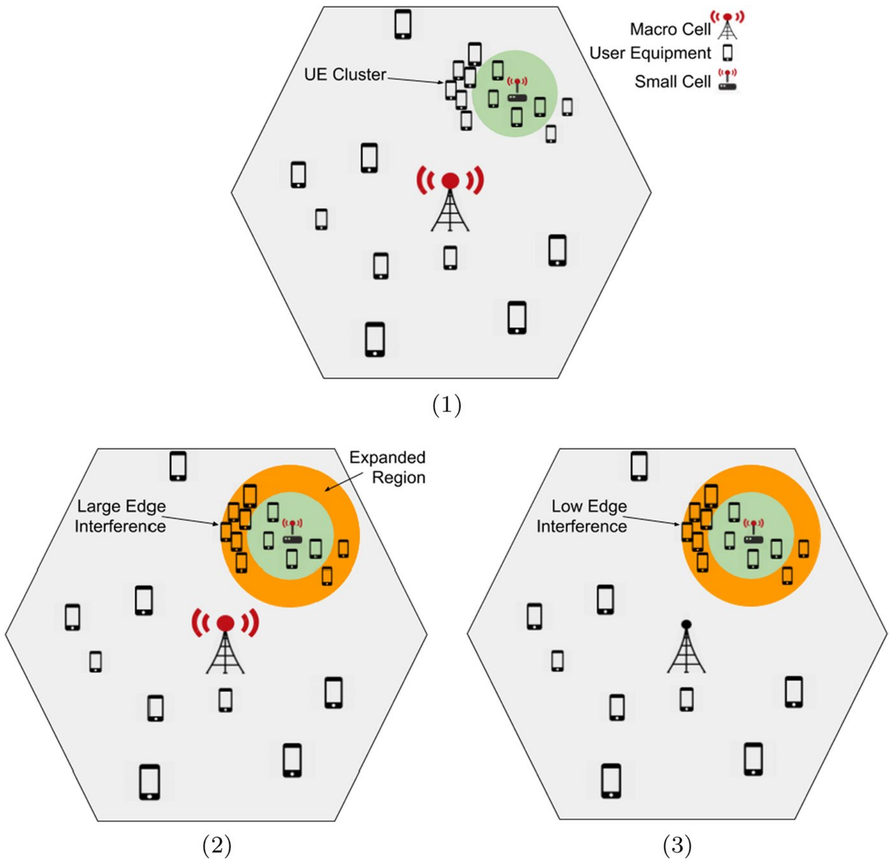


Fig. 1 Example of a heterogeneous network with one macro cell, one small cell, and over 20 user equipments. Part **1** shows few UEs in the reach of the SC, while the rest of UEs are attached to the MC. Therefore, clustered UEs experience high interference and poor connectivity. Part **2** shows the SC expanded region which allows the SC to attach clustered UEs and improve their connectivity. However, this leads to a large interference at the edge of the expanded region between the MC and the SC. Part **3** shows the muting of MC which lowers the interference at the edge of the SC expanded region. However, reducing the communication time of UEs attached to the MC at the same time

attached to MCs can no longer receive data from their respective cells during that period.

Figure 1 shows a HetNet example with one MC, one SC and 21 UEs. The upper part (i.e., part 1) shows that the SC is not powerful enough to attach the clustered UEs. Therefore, these UEs are attached to the MC, creating a hot-spot with a high interference and a poor connectivity. In the lower left part (i.e., part 2), the SC uses the signal bias to expand its region to reach the clustered UEs, thus, improving the connectivity of the clustered UEs and reducing the load on the MC. However, this creates a severe interference at the edge of the expanded region. The lower right part (i.e., part 3) shows the muting of the MC which mitigates drastically the interference at the edge of the SC expanded region.

2.2 Scheduling in heterogeneous networks

The downlink rate R_i^f is a measure (in bit per second) of the amount of data that a UE u_i is able to download during the SF S_f. R_i^f is commonly described in the literature using Shannon's formula [12]. The downlink rate depends on: (1) the bandwidth B, (2) the number N_f of UEs communicating during the same SF S_f and (3) the Signal to Interference and Noise Ratio (SINR):

$$R_i^f = \frac{B}{N_f} \times log_2(1 + \text{SINR}_i^f) \tag{4}$$

On one hand, the high signal of MCs makes UEs attached to them experience high Signal to Interference and Noise Ratio. Therefore, conferring UEs attached to MCs high downlink rates whenever the MC is active (i.e., not muted) and not overloaded. Hence, all the UEs that are attached to MCs could be scheduled for transmission during all SFs at which the MCs are active, making their scheduling trivial. On the other hand, SCs are low powered devices, making UEs that are attached to them experience a relatively weak signal. Additionally, UEs attached to SCs would be subject to a large interference by MCs during their active SFs.

The bandwidth is known to be an expensive and a scarce resource, which makes it hard to improve. However, we could aim to improve the two remaining elements: (1) the SINR and (2) the number of transmitting UEs N_f. We could improve significantly the SINR_i^f by simply muting MCs at more sub-frames of a given frame $\mathcal{F}$. Despite the possible improvement in the signal to noise ratio, exaggerating the muting of MCs would substantially reduce the overall downlink rate of their attached UEs as they would not receive any data in the meantime. Additionally, the number of UEs attached to MCs are often more numerous. Therefore degrading their connectivity and worsening significantly the overall performance of the HetNet. Similarly, we could also reduce the number of UEs scheduled for communication at the same SF. Fewer communicating UEs would improve the downlink rate for the UEs that are scheduled for transmission. However, it would also mean that non-scheduled UEs would not be receiving any data.

The combination of all these aspects makes scheduling transmissions in a HetNets a non-trivial problem, requiring an autonomic scheduling system. The scheduling system would have to define the SFs when MCs are active or muted, in addition to the scheduling of communication between UEs and their respective cells (most importantly SCs) at each SF.

2.3 Evaluating the quality of a transmission schedule

Every UE $u_i \in \mathcal{U}$ experiences an average downlink rate $\bar{R}_i$ over all SFs of the same frame such that:

$$\bar{R}_i = \frac{1}{|\mathcal{F}|} \sum_{S_f \in \mathcal{F}} R_i^f \tag{5}$$

Operators of HetNets commonly aspire to generate schedules that optimise fairness of downlink rates experienced by the various UEs connected to their network [13]. This is also the aim of the state-of-the-art work [10] against which we are comparing to in our paper. The fairness in downlink rate could be expressed as the sum of average downlink logs (i.e., $log(\bar{R}_i)$) of the various UEs:

$$Fairness = \sum_{u_i \in \mathcal{U}} log\left(\bar{R}_i\right) \tag{6}$$

We assume in our work that the quality of the transmission schedules is a good estimate for the quality of the scheduler that generated them. Therefore, we consider that the quality of a scheduler is equal to the fairness of its generated schedules.

In our work, we aim at generating a scheduler that schedules communications $M[j][i][f]$ between the cells $c_j \in \mathcal{C}$ and UEs $u_i \in \mathcal{A}_j$ at every sub-frame s_f, while maximising the fairness in Eq. 6 as the fitness function. The scheduler set $M[j][i][f]$ to 1 to allow the communication between the cell c_j and the UE u_i at the sub-frame s_f, or 0 otherwise. Improving the logs of average downlinks is highly affected by UEs with low average downlink rates. At the same time, it does not reward UEs with excessively high average downlink rates.

3 Related work

In this section, we first review the work related to the use of genetic programming (GP) in scheduling, then we study the optimisation techniques that were brought to scheduling in heterogeneous networks from the simple heuristic- (rule-) based ones to the more recent evolutionary techniques like grammar-based GP. Last, we review different works on developmental evaluation and layered learning in GP and explain their difference with our proposed multi-level grammar approach.

3.1 Genetic programming in scheduling

There exist multiple scheduling problems that arise across various application domains [14] such as timetabling, cloud computing and traffic control. While the research usually explores the search space through enumeration of the solutions using metaheuristics, these solutions are rarely applied in real production environments as they are slow and cannot perform the optimisation in a timely manner.

Jakovocic and Marasovic [15] showed the shortcomings of both enumeration-based and search-based approaches to tackle scheduling problems where the execution time is limited. The authors put forward meta-algorithms that were manually designed to handle the specific job-shop scheduling environment at hand. The meta-algorithms also come with priority functions that are evolved using genetic programming techniques to operate within them. Therefore, the structure of the solution is informed by the domain knowledge brought by the authors while designing their meta-algorithms. Whereas the genetic programming technique captures the underlying complex mappings of statistical features to fitness functions. Branke et al. [16] put an emphasis on the effort and care that need to be taken when designing the meta-algorithm to fit the given problem. The authors also recognise the value of meta-algorithms and automated techniques in general given that the design of heuristics to optimise specific problems is also a hard task.

Genetic programming has since been applied on numerous scheduling problems [17]. Nguyen et al. [17] argue that genetic programming is a powerful technique for an automated design of production scheduling heuristics. The authors claim that in addition to outperforming many manually designed heuristics in the literature, they also discover very sophisticated ones, thus dealing with more dynamic and more complex environments. Nguyen et al. developed a unified framework under which they then survey and compare the literature on automating the design of scheduling heuristics using genetic programming.

3.2 Scheduling in heterogeneous networks

The most recurrent works in the literature of scheduling in heterogeneous networks hand-craft the algorithm design by expert agents. Most employed techniques partition UEs that are attached to SCs into two different groups [13, 18, 19] according to sub-frames (i.e., a standardised time window) they are scheduled to communicate at: (1) ABS sub-frames: sub-frames during which MCs are forbidden from transmitting or (2) non-ABS sub-frames; sub-frames during which all MCs are allowed to transmit/receive data from their attached UEs.

Weber and Stanze [13] designed two different schedulers by hand: (1) strict scheduler: where all UEs at the centre of the cell are sacrificed during all the sub-frames that are flagged as either ABS or protected and (2) dynamic scheduler: where UEs at the edge of the cell are allowed to transmit/receive data regardless of the flag of the sub-frame being ABS or non-ABS. Strict scheduler provides UEs at the edge of cells with an improved bandwidth when having the highest channel quality, but mutes these cells during non-ABS sub-frames. The authors showed experimentally that a better trade-off between download rate and spectral efficiency is achieved by the dynamic scheduler at UEs in cell edges. Pang et al. [18] put forward a distributed scheduling method that determines through dynamic programming which UEs to protect by ABS for a given number of frames. Their technique also finds the best number of ABS sub-frames through an evaluation of the global system utility. Jiang and Lei [20] model the scheduling problem as a two-player bargaining game seeking to find a Nash equilibrium. The game is organised between ABS and Non-ABS sub-frames with each of them trying to attract UEs in order to transmit in a given time interval. Lopez et al. [19] place UEs into either an overlapping queue or a non-overlapping queue according to their mean Signal to Interference and Noise Ratios. The algorithm repeatedly identifies UEs with the lowest average Signal to Interference and Noise Ratio in each queue, estimates the queue sizes that would balance the rates between the worst performing UEs, before transferring UEs between queues. This process is repeated until a balancing of the downlinks in both queues is achieved. The proposed heuristic improves the 5th percentile downlink by 55% for UEs attached to small cells in comparison to the baseline. Finally, Deb et al. [21] model the scheduling problem as a Mixed-Integer Non-Linear Program and develop a novel algorithm to solve it with a provable guarantee of optimality. Their approach takes into account the network traffic load, the topology, and the interference in the Small and the Macro cell range.

Fagan et al. [22] propose a hybrid algorithm (i.e., Deep Learning Through Evolution) which combines a deep learning technique with an evolutionary algorithm. The authors use a genetic algorithm to generate a set of optimised solutions (human-competitive solutions). Then, they use a deep learning technique (i.e., Deep Feed Forward Neural Network with 4 hidden layers) to learn the model underlying within the solutions. The learned neural net allows the authors to achieve a similar performance as the genetic algorithm while only requiring almost a 100 times less execution time. Despite the performance of the learned neural network and its fast execution, the approach proposed by Fagan et al. [22] suffers from the excessive learning time (i.e., several runs of an already expensive Genetic Algorithms, combined with time-consuming deep neural networks).

3.3 Genetic programming for scheduling in heterogeneous networks

Genetic programming has been applied on various scheduling problems [17] from different domains. Furthermore, genetic programming has also been used in heterogeneous networks prior to the work of Lynch et al. [10]. However, it has not been applied specifically on the scheduling of transmissions. For instance, Ho et al. [23] apply genetic programming in order to improve the coverage during the deployment of femtocells[1] in industrial companies. Their study represented a proof of concept that it is possible to automatically evolve controllers for wireless networks. Hemberg et al. [24] also aim at optimising femtocell coverage in heterogeneous networks. The authors use a Grammatical Evolution algorithm to evolve symbolic expressions that are then mapped to solutions that outperform humanly designed heuristics on two out of three objectives.

There are only a limited and fairly recent number of works in the field of autonomic scheduling in heterogeneous networks which use genetic programming techniques to design and learn interesting scheduling heuristics. Motivated by the claims of Dempsey [25] that genetic programming methods yield robust solutions in dynamic environments, Lynch et al. [10] are the first to apply a version of genetic programming (i.e., a grammar-guided genetic programming) for the scheduling in heterogeneous networks. Their technique evolves individuals (i.e., expressions) that are then mapped to transmission schedulers. The mapped schedulers are then

[1] Femtocells are small cells with only few meters in tranmission range.

deployed in a HetNet to generate transmission schedules in a millisecond timescale. The authors showed that their approach successfully evolves robust human-competitive schedulers. Additionally, the authors claim that their approach can automatically discover good solutions and could even tackle corner cases (i.e., finely tune their algorithm to cities with specific HetNets that are different from the standard ones) providing the use of an appropriate fitness function. The authors also apply a genetic algorithm with no time restriction (at least not in a millisecond) and showed the existence of large potential improvements. Lynch et al. [26] expand on their initial work [10] and compare two different fitness functions: (1) evaluative feedback: the industry standard utility heuristic for downlink rates and the state-of-the-art and (2) instructive feedback: which is obtained through an offline optimised scheduler learned using genetic programming and acting as a semantic for evolving models. The authors also compare two approaches for mapping trees to schedules and show that their approach outperforms the previous industry state-of-the-art.

We have recently proposed a multi-level approach for G3P [11]. We designed multiple grammars—each with its own granularity (i.e., small, medium and full). We showed that the two-level G3P approach (lower level: running G3P with the small grammar for a few generations, and upper level: introducing and continuing the evolution with the full grammar) outperforms the classical G3P (i.e., with a unique grammar from the beginning to the end) on the Scheduling in HetNets. In our current work, we extend [11] as we identify the shortcomings of the taken strategy (using all the last population of evolved individuals at the lower level, as an initial population of individuals in the evolution at the upper level). We also propose an alternative strategy (i.e., seeding of the best individual from the lower level to a newly generated population in the higher level) to cope with them and evolve better scheduling expressions. Recently, we have evaluated the use of a hierarchical grammar strategy to the multi-level G3P [27]. We start our algorithm with multiple runs of G3P with small grammars (each run with a different small grammar), before aggregating the best individuals and continuing the evolution with the full grammar. This new strategy has shown, providing an adequate number of small grammars at the lower level, to yield better scheduling expressions. However, this work also uses a population porting strategy.

Fenton et al. [7] also apply a genetic algorithm approach to design heuristics for the scheduling in heterogeneous networks. However, they consider an environment where the knowledge is partial and that the algorithm is not fully aware of the performance at each UE. The last work from Fenton et al. [2] uses genetic programming for optimising not only the scheduling in heterogeneous networks but also the power setting and the bias selection at the small cell level, in addition to the management of the macro cells duty cycle.

3.4 Multi-level learning

Optimisation techniques commonly use hybridisation as a multi-level strategy to improve their results (either metaheuristics with metaheuristics [28, 29] or metaheuristics with exact solvers [30, 31]). However, the majority of evolutionary algorithms are monolithic. There exist few works that challenge this monolithism in evolutionary algorithms (and GP in particular) through various strategies. In this part, we try to summarise the most-known amongst them in four categories: (1) developmental evaluation, (2) layered learning, (3) transfer learning and (4) the newly introduced multi-level grammars.

3.4.1 Developmental evaluation GP

McKay et al. [32] are the first to investigate what they called the developmental evaluation of genetic programming. The authors propose to use a different and more complex fitness at every generation, while simultaneously evaluating a larger part of the tree based genotype—without modifying it in any way though. The same authors argued in [33, 34] that it is necessary to perform the evaluation during the development to allow the emergence of structural regularities. The authors expand on their initial developmental evaluation and propose a new representation called Developmental Tree Adjoining G3P (DTAG3P [34]). Their DTAG3P representation encodes tree adjoining grammar (TAG) derivation trees using L-systems. The authors showed that their approach could improve the performance of a regular GP on benchmark problems e.g., polynomial symbolic regression, Fourier series fitting and parity problems [35], but not on many other ones. Furthermore, despite the improvement, DTAG3P is not directly comparable to a standard GP as it requires more domain knowledge in the design of the developmental fitnesses.

McPhee et al. [36] also propose a developmental evaluation approach to linear-GP style programs by expanding the Estimation of Distribution Algorithm (i.e., N-Gram GP systems) with an Incremental Fitness-based Development (IFD).

3.4.2 Layered learning GP

Layered learning is similar to developmental evaluation in the way that it creates intermediary fitness functions. The relationship between the two approaches has already been investigated by Hoang et al. [37]. Layered learning differs from developmental evaluation as only a few and finely designed fitness functions are used. Furthermore, in layered learning, the evolutionary algorithm is allowed a larger computational cost (i.e., more generations) in order to improve its fitness with regards to the given fitness function. In other words, in layered learning, the problem is decomposed into few sub-problems. GP is then applied to the sub-problems one after the other while porting the population resulting of a sub-problem as an initial population to the next one.

Gustafson and Hsu [38] are the first to propose a layered leaning GP algorithm to evolve agents capable of playing keep-away soccer. Their layered learning approach allowed them to create better agents than those created by a standard GP. Moreover, their approach was also faster. Jackson and Gibbons [39] proposed a two-layered approach in order to optimise basic problems (i.e., Boolean logic, even-parity and majority) and showed that their approach outperforms a standard GP. Hien et al. [40] combine both layered learning and incremental sampling in GP to address a symbolic regression problem. The authors showed that their approach outperforms the standard GP on twelve benchmark problems and is faster. Despite this improvement, the authors do not compare their approach to the basic layered approach and do not justify the added value of incremental sampling. Even worse, the sampling adds extra parameters that need to be tuned. Afterwards, Hien and Nguyen [41] extend their work to include a parameter setting technique inspired by progressive sampling, thus coping with their hyper-parameters issue.

3.4.3 Multi-level grammars-guided GP

We have recently proposed a novel multi-level grammar approach to grammar-guided genetic programming [11]. Unlike what has been done with the two previous approaches (i.e., developmental evaluation and layered learning GP), we do not design multiple fitness functions, but rather design different grammar levels.

The main difference is that we do not modify the fitness functions at the different levels. We have the same fitness function for all the grammars, thus we maintain the shape of the fitness landscape when passing from a grammar to another.

Grammars are designed with different complexities/granularities in terms of terminals. Therefore, evolution with a less restricted grammar can only generate individuals with a small set of functionalities (terminals). Therefore, while evolution with a small grammar navigates the same search space as evolution with a full grammar, it can only navigate parts of it (parts that can be navigated with the limited number of terminals).

We use a small grammar for a certain number of generations to evolve interesting individuals in a restricted search space. Then, we add more functionality at different grammar levels to increase the search space and allow the evolution to evolve individuals with a better fitness.

3.4.4 Transfer learning and reuse of extracted knowledge GP

While the aforementioned approaches are all embedded within a single evolutionary process, transfer learning and the reuse of extracted knowledge happen between multiple evolutionary processes of different problems or even different domains.

Transfer learning is the application of skills and knowledge learned from a former problem domain to a novel one. While transfer learning is well-known and used in many areas of machine learning such as image analysis, it did not attract many works in GP [42, 43].

Dinh et al. [42] put forwards some transfer learning techniques for GP whereby they transfer a number of good quality individuals or part of them from the known problems to the new ones. The authors showed on two symbolic regression problems that their methods help GP to achieve smaller training errors. The authors also showed a performance improvement over GP when dealing with validation data. Moreover, the authors noticed that transferring individuals of a limited size reduces bloat and limits code growth. Haslam et al. [43] investigate in more details the performance of the methods proposed by Dinh et al. [42]. More particularly, they analyse the influence of key parameters on performance. They also study the effect of transfer learning on the evolution process. They show that transfer learning provides the GP in the new problem with a good initial population. This allows the learning to speed up the convergence while achieving a better performance. However, the authors acknowledge that the benefits of transfer learning vary from a scenario to another. Iqbal et al. [44] push the boundaries further as they use transfer learning to help GP address the more complex problem of classifying noisy and rotated images. The authors designed a mechanism to automatically discover, extract and reuse blocks of knowledge and information from similar image classification problems and injecting them into the problem at hand. In their proposed approach, the selection of interesting information consists of sub-trees common to the different problems. They showed that their solution outperforms the state-of-the-art classifier while GP alone could not achieve any satisfactory performance.

4 Multi-level grammar-guided genetic programming

In this section, we describe both the state-of-the-art grammar-guided genetic programming algorithm for scheduling in HetNets and our multi-level grammar-guided genetic programming approach.

4.1 State-of-the-art: grammar-guided genetic programming

Lynch et al. [10] designed a Grammar-Guided Genetic Programming algorithm for the scheduling in HetNets. The state-of-the-art algorithm G3P is an adaptation of a grammar-based form of GP [45] as implemented in the PonyGE 2 framework [46]. The algorithm proposed by Lynch et al. evolves an expression in a tree form. Each evolved expression maps the Signal to Interference and Noise Ratio related statistics and attachment information of the UEs $\mathcal{U}$ and cells C to transmission scheduler using Algorithm 1 which generates a binary decision for each UE per SF. The binary decision defines whether or not to schedule the UE and allow it to transmit to its attached cell (macro or small cell) at a given SF.

Lynch et al. [10] use a grammar in a Backus-Naur Form (BNF) to incorporate domain knowledge to their algorithm. The authors use a grammar that is thorough in terms of networks-related statistics, from the beginning to the end of the G3P process:

```
<expr> ::= <reg> | <reg> | <reg> | <Terminal>
<reg> ::= <expr><op><expr> | <expr><op><expr> | <expr><op><expr> |
   <expr><op><expr> | <non-linear>(<expr>) | <non-linear>(<expr>)
   <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_viable | 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
```

Table 1 Description of the network communication statistics used as terminals in the grammar of the state-of-the-art G3P

Statistic	Description
downlink	Amount of data (bit/s) that could be transferred in a unit of time $log_2(1+\mathrm{SINR}_i^f)$
num_variable	The noise is too high for a UE to communicate $\mid\{\mathrm{SINR}_i^f \geq 1\}\mid_f$
num_att	Number of UEs that are attached to the Small Cell $\lvert\mathcal{A}_j\rvert$
airtime	Number of sub-frames at which a given UE is allowed to transmit $\mid\{S_f \in \mathcal{F} \mid M_j[j][i][f]=1\}\mid$
congestion	Number of UEs transmitting at the same sub-frame $\mid\{u_i \in \mathcal{U} \mid M[j][i][f]=1\}\mid$
avg_downlink_frame	Average downlink of a UE over all sub-frames $\frac{1}{\mathcal{F}}\sum_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)$
max_downlink_frame	Maximum downlink of a UE over all sub-frames $\max_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)$
min_downlink_frame	Minimum downlink of a UE over all sub-frames $\min_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)$
avg_downlink_SF	Average downlink of UEs attached to a SC at a given sub-frame $\frac{1}{\mathcal{A}_j}\sum_{u_i\in\mathcal{A}_j} log_2(1+\mathrm{SINR}_i^f)$
max_downlink_SF	Maximum downlink of UEs attached to a SC at a given sub-frame $\max_{u_i\in\mathcal{A}_j} log_2(1+\mathrm{SINR}_i^f)$
min_downlink_SF	Minimum downlink of UEs attached to a SC at a given sub-frame $\min_{u_i\in\mathcal{A}_j} log_2(1+\mathrm{SINR}_i^f)$
avg_downlink_cell	Average downlink per cell of average downlink per sub-frame $\frac{1}{\mathcal{A}_j}\sum_{u_i\in\mathcal{A}_j}\left(\frac{1}{\mathcal{F}}\sum_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)\right)$
max_downlink_cell	Maximum downlink per cell at any given sub-frame $\max_{u_i\in\mathcal{A}_j}\left(\frac{1}{\mathcal{F}}\sum_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)\right)$
min_downlink_cell	Minimum downlink per cell at any given sub-frame $\min_{u_i\in\mathcal{A}_j}\left(\frac{1}{\mathcal{F}}\sum_{S_f\in\mathcal{F}} log_2(1+\mathrm{SINR}_i^f)\right)$

Most production rules in the aforementioned grammar are well-known to the GP community and are easy to understand. However, <statistic> contains terminals that are from the communications network domain. Despite not being important for the understating of this work, we describe briefly each of them in Table 1.

Lynch et al. [10] use Algorithm 1 to do the mapping from tree expressions to transmission schedulers, which are then used to generate transmission schedules, before evaluating their fitness function using Eq. 6. Their approach evaluates the expression as many times as they have to decide whether to schedule a UE to transmit at a given sub-frame. In addition to the UE and the sub-frame, the evaluation of the expression is joined with the current state of the network (i.e., before scheduling the current UE). The evaluation process returns a value of interest out of which it is decided to allow the UE to communicate or not. A UE is set scheduled to communicate providing a positive interest and an ability to communicate (i.e., noise is not too large).

Note that the fitness of each individual (i.e., expression) that is evolved by G3P is computed as the Fairness (Eq. 6) of the transmission schedules that are obtained by the scheduler that they map to using Algorithm 1.

```
Parameters : $E$: Expression
input      : $\mathcal{C}$: Cells, $\mathcal{U}$: UEs
output     : $M$: Schedule Matrix
for $c_j \in \mathcal{S}$ do
    $M[j] \leftarrow zeros(|\mathcal{A}_j| \times |\mathcal{F}|)$ // define a matrix of 'not scheduled'
    for $S_f \in \mathcal{F}$ do
        for $u_i \in \mathcal{A}_j$ do
            $interest \leftarrow evaluate(E, M[j], i, f, \mathcal{C}, \mathcal{U})$ // evaluate expression for $u_i$
               in f with current Schedule M
            if $interest > 0$ and $SINR_i^f \geq 1$ then
                $M[j][i][f] \leftarrow 1$ // set as 'scheduled'
            end
        end
    end
end
return $N$;
```
Algorithm 1: Mapping of a tree expression to a transmission schedule.

4.2 Multi-level grammar approach

In our multi-level grammar approach, we also keep the full and more thorough grammar (i.e., F: full) that was defined by the state-of-the-art. Moreover, we define two new grammars (i.e., S: small and M: medium) through the modification of the two production rules which contain only terminals (i.e., <const> and <statistic>). The two grammars are created by restricting and increasing the list of available terminals at each grammar in an incremental fashion, so that S is included in M and M is included in F.

In the small grammar S, we restrict and reduce the number of terminals to the minimum. We only keep a small list of well-spread constants (i.e., 0, 0.5 and 1). We also only keep a selection of the most critical communications network statistics. The downlink is what we aim to optimise through our work. On the other hand, improving the min_downlink_frame is likely to drive the smallest downlinks up, and thus lead to an improvement in fairness (i.e., improve the fitness function).

```
<const> ::= 0.0 | 0.5 | 1.0

<statistic> ::= downlink | min_downlink_frame
```

We also define the medium grammar through an increment of six terminals to the <const> and <statistic> production rules from the previously defined small grammar. We add four constants (two signs × two constants) that are spread in a dichotomic way to the <const> production rule. We also add to the <statistic> production rule two communications network statistics (i.e., max_downlink_frame and min_downlink_cell) that are also closely related to the downlink.

```
<const> ::= 0.0 | 0.3 | 0.5 | 0.8 | 1.0

<statistic> ::= downlink | min_downlink_frame | max_downlink_frame |

   min_downlink_cell
```

In conjunction with the definition of the three grammars (i.e., S, M and F), we extend the state-of-the-art G3P algorithm, by making it capable of taking one grammar at the start of the experiment and dynamically updating its grammar to more complex ones (e.g., from S to M, M to F, or S to F). All individuals obtained using a given grammar are ported as an initial population [47] 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 interpretation (in terms of schedule) both before and after changing the grammar. Given that we do not modify the fitness function during the evolution, the individuals also have the same fitness before and after updating the grammar. Therefore, the multi-level grammar approach does not add any computational cost to the regular G3P algorithm (with the exception of the computationally cheap grammar substitution).

Note that we do not update any of the evolutionary parameters (e.g., mutation rate, crossover rate, etc.) while modifying the grammar. We take this decision in order to avoid any impact that changing the values of these parameters could have. Thus, we only leave one varying element at a time (i.e., either the grammar or the number of individuals that are kept for the evolution after introducing a new grammar).

Figure 2 shows an example of the multi-level grammar approach with a population porting strategy. G3P randomly generates an initial population using the small grammar S. The initial population is then evolved using that same grammar for a certain number of generations to obtain the last population with S. Then, G3P

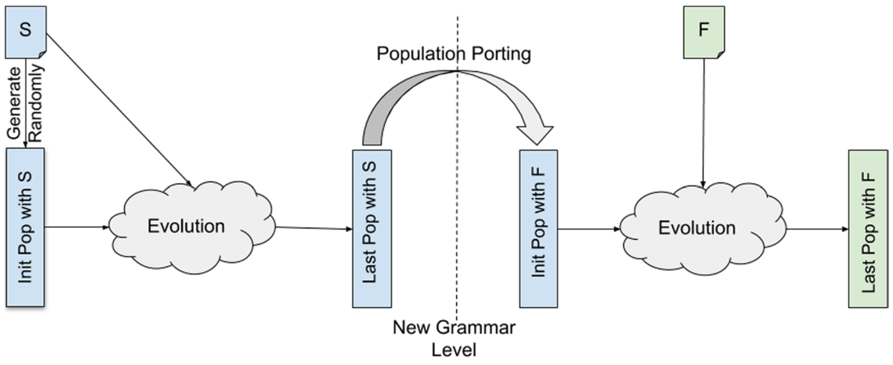


Fig. 2 Example of the multi-level grammar approach with the population porting strategy. G3P generates individuals using the grammar S and evolves them for a few generations using that same grammar. Then, G3P takes the population that resulted from the evolution with S and evolves it further using the grammar F

substitutes the grammar from S to F. However, instead of generating a new population, the last population with S is used as an initial population with F without any modification. Last, G3P evolves its population using the grammar F to obtain a last population (in this case, the last population for the entire evolution process).

5 Experiment design

In this section, we briefly describe the dataset and the setup we use to perform our experiment. Furthermore, we describe the statistical test that we use to assess the significance of our evaluations.

5.1 Dataset

We use in our experiment three scenarios corresponding to three different simulated HetNets which were generated following the same process as in [10, 26]. All three scenarios cover the same geographical area that encompasses the 3.61 km^2 of Dublin city centre shown in Fig. 3. MCs are placed on a grid by network operators, while SCs are typically deployed in an ad-hoc fashion to serve hotspots. Therefore, we arrange MCs in a hexagonal pattern and we randomly scattered SCs on the map. We also randomly place UEs on the map in the simulated HetNets. Note that in our simulations, UEs occupy static positions and transmit whenever they are scheduled. While this assumption might not be realistic for long simulations, it is accurate enough given the millisecond time-frame at which our algorithms must work (our algorithms have to create a transmission schedule that is best for the current environment in milliseconds).

All the HetNets have exactly 21 MCs that are distributed in a hexagonal pattern. However, they contain a different number of SCs each. The first HetNet has 21 SCs and is the scenario with the smallest density (1 SC per MC on average). The second

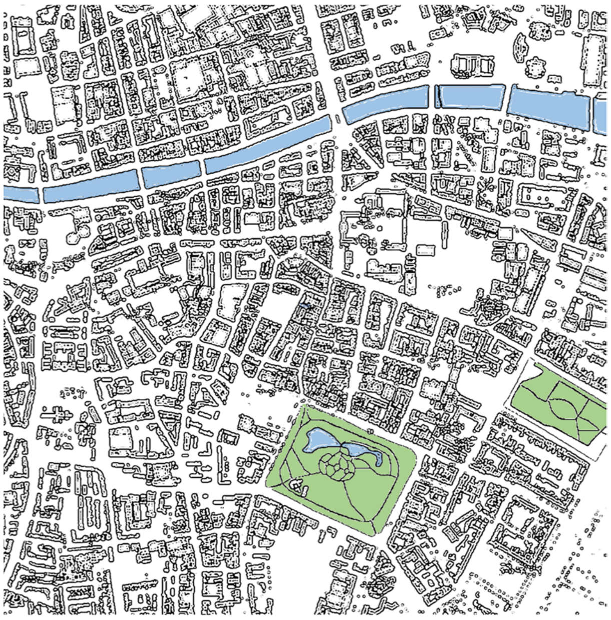


Fig. 3 The 3.61 km^2 area of Dublin city centre that is covered in our experiment. Streets and town squares are in white. Buildings are sketched boxes. Green parks are in green and water points are in light blue (Color figure online)

HetNet has 63 SCs and is three times denser than the previous HetNet (3 SCs per MC on average). The third and last HetNet has 105 SCs and is the densest amongst the different HetNets (5 SCs per MC on average). Moreover, the three scenarios similarly scatter 1250 UEs in the covered geographical area and attach each of them to the cell with the largest signal strength plus bias at their location (i.e., either MCs or SCs).

Note that all MCs, SCs and UEs are statically placed on the used map of Dublin city. Furthermore, the signal strength and noise that are experienced by UEs, while being realistic (dependent of the environment), are not stochastic. This enables us to analytically evaluate the fitness of an evolved expression.

5.2 Setup

We use in our experiment the state-of-the-art G3P algorithm that is put at our disposal by its authors. Furthermore, we define the various evolutionary parameters in the same way as in [11]. We set the size of the population to 100 and run the algorithm for 100 generations. Moreover, we generate the initial population of individuals with a maximum tree depth of 20, using the ramped half–half (RHH [48]) algorithm. We use the sub-tree crossover with a probability 0.7 and undergo a sub-tree mutation once for every individual. We set all the other parameters as shown in

Table 2 Evolutionary parameters used for both the state-of-the-art G3P and the multi-level G3P

Initialisation	Ramped half–half
Max initial tree depth	20
Overall max tree depth	20
Population size	100
Number of generations	100
Selection	Tournament
Tournament size	10% of population
Replacement	Generational with elites
Elite size	1% of population
Crossover type	Sub-tree with a 70% probability
Mutation type	Sub-tree once per individual
Number of runs	100

Table 2. Additionally, we repeat every experiment 100 times to minimise the effect of randomness.

5.3 Significance

The inspection of the results obtained from each of our algorithms over the multiple runs (i.e., in our case, 100 runs) do not follow a normal distribution (assessed using the Shapiro-Wilk normality test [49]) and do not have a homogeneous variance (assessed using Levene's test of homogeneity of variances [50]) in most cases. Therefore, we perform a statistical significance test using a non-parametric test: the two-tailed Mann–Whitney U test (MWU). MWU is given the different performance values (best fitness) achieved by two algorithms at a time from each run for a particular experiment case. MWU returns the p value that one of the algorithms obtains different values than the other. We also consider the standard Bonferroni adjustment [51] in order to reduce threats of having type I errors (rejecting a true null hypothesis incorrectly in case of multiple comparisons). Therefore, we consider that our tests are significant when their p value is below 5%.

Furthermore, we measure the substantive significance using Cohen's d Effect Size [52] between two given sets of performance values. The Effect Size refers to the magnitude of the difference between the two sets, which corresponds to the difference between their mean outcomes. As general guidelines, the effect size is considered small, medium and large when its value is respectively larger than 0.2, 0.5 and 0.8.

6 Multi-level grammar through population porting

In this section, we would like to confirm that using the multi-level grammar approach with the population porting strategy could improve the performance of the grammar-guided GP algorithm. Furthermore, we investigate the sensitivity of the approach to the phase at which grammars are introduced, before delving into the reasons that lead to this sensitivity.

6.1 Performance of the multi-level grammar approach with population porting

In order to show the relevance of combining different grammars, we compare 6 grammar configurations on the three instances (21 SCs, 63 SCs, and 105 SCs). We designed 6 different grammar configurations:

- S: use the small grammar from the beginning to the end.
- M: use the medium grammar from the beginning to the end.
- F: use the full grammar from the beginning to the end.
- S10M: start with S and introduce M at generation 10.
- S10F: start with S and introduce F at generation 10.
- S10M20F: start with S and introduce M and F at generations 10 and 20, respectively.
- M10F: start with M and introduce F at generation 10.

In the configurations which use a succession of grammars (i.e., S10M, S10F, S10M20F and M10F), the entire population that is obtained from the evolution process using the small grammar is 'ported' as an initial population to the evolution with the larger grammar when performing the update of the grammar.

Figure 4 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 100 runs).

We see from Fig. 4 that the algorithm G3P improves the best fitness function in all instances with each of the grammar configurations it is applied with (G3P constantly improves its initial best performance). We also see that limiting the number of generations to 100 does not seem to allow to achieve a full convergence for the various algorithms. Therefore, increasing the number of generations is likely to yield a better overall performance—but necessitates a larger computational cost.

We see that starting with the small grammar and introducing the full grammar at generation 10 (i.e., S10F) achieves the best results in all instances. This shows clearly the importance of using a multi-level grammar strategy rather than a full one from the beginning to the end. This behaviour is somewhat surprising when we look at the shape of the curve. S10F starts to converge to the worst results around generation 10, before drastically improving its performance and outperforming all the other configurations. This indicates that the algorithm takes advantage of the small grammar to evolve individuals that are not interesting from the fitness point of view, but that are interesting to the evolutionary process. S10F is followed by the configuration which uses only the full grammar (i.e., F) and indicates that while not being the most performing configuration, F achieves good results nonetheless. While S10F outperforms slightly F on the instance 21 SCs, the margin increases on 63 SCs and 105 SCs.

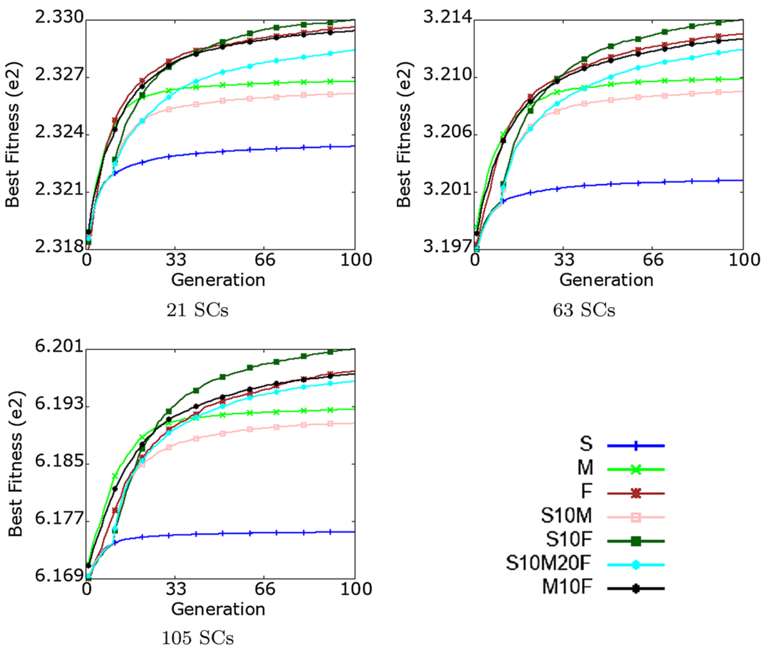


Fig. 4 Mean over 100 runs of the evolution of the best fitness obtained by G3P on the different instances (i.e., 21, 63 and 105 SCs) using various grammar configurations (i.e., S, M, F, S10M, S10F, S10M20F and M10F)

Starting with the medium grammar and introducing the full one at generation 10 (i.e., M10F) has a quasi-similar behaviour as F. This shows that starting and using the medium grammar for 10 generations does not help the optimisation as much as with the small grammar. We should note that using M for the first 10 generations does not act as a handicap either. Unlike, in generation 10 of S10F, we do not see any inflection point. This could be explained by either (1) the early introduction of the full grammar, or (2) the medium grammar is large enough to cover the lack of elements from the full grammar. However, looking at the curve with only the medium grammar, we see that using M alone achieves way worse results than F. This allows as to discard the first hypothesis and confirm that the lack of inflection point in M10F is likely due to the early introduction of the full grammar. Therefore, if we increase the phase at which the grammar F is introduced (e.g., M20F or M30F) we are likely to start noticing that inflection point as well.

Starting with the small grammar and using the medium instead of the of the full one at generation 10 (i.e., S10M) does not perform as well as S10F. S10M performs actually worse than M alone. In addition to the fact that M is not broad enough as a grammar to allow a convergence to good individuals, the configuration S10M suffers as well from the time/computation wasted using the small grammar. Despite having the same inflection point as S10F, the improvement that is obtained after introducing the medium grammar is not as intense as with the full grammar.

Table 3 Mean, min, max and standard deviation over 100 runs of the best fitness obtained by G3P when using different grammar configurations (i.e., S, M, F, S10M, S10F, S10M20F and M10F). In addition, we include the *p* value (Mann–Whitney U test with Bonferroni adjustment) and the effect size in comparison to the results obtained against G3P with the grammar F. We put in bold the best mean, min, max, significant *p* values and positive medium/large effect sizes (i.e., larger than 0.5)

Instance	Metric	S	M	F	S10M	S10F	S10M20F	M10F
21 SCs	Mean	232.334	232.673	232.957	232.609	**232.995**	232.837	232.937
	Min	232.007	232.658	**232.880**	232.208	232.868	232.414	232.392
	Max	232.401	232.695	233.040	232.693	233.102	233.127	**233.184**
	Sd	0.0516	0.0093	0.0509	0.0712	0.0627	0.1621	0.1447
	p value	**3.22E−25**	**4.36E−20**	–	**2.90E−25**	**3.10E−03**	**3.80E−06**	1.00E+00
	Effect size	− 12.044	− 8.374	–	− 5.206	**0.647**	− 0.844	− 0.155
63 SCs	Mean	320.200	320.986	321.334	320.888	**321.443**	321.211	321.296
	Min	320.037	320.934	321.206	320.416	**321.238**	320.428	320.610
	Max	320.290	321.031	321.467	321.033	321.681	321.675	**321.691**
	Sd	0.053	0.026	0.075	0.118	0.131	0.267	0.270
	p value	**3.22E−25**	**4.36E−20**	–	**3.98E−25**	**4.73E−05**	**1.25E−02**	1.00E+00
	Effect size	− 18.807	− 6.542	–	− 4.103	**0.971**	− 0.525	− 0.162
105 SCs	Mean	617.583	619.273	619.799	619.080	**620.101**	619.661	619.760
	Min	617.314	619.179	619.537	618.237	**619.708**	618.009	618.326
	Max	617.635	619.382	620.107	619.370	620.598	620.557	**620.718**
	Sd	0.038	0.058	0.160	0.241	0.244	0.531	0.530
	p value	**4.94E−25**	**4.36E−20**	–	**3.58E−25**	**1.93E−09**	4.60E−01	1.00E+00
	Effect size	− 24.688	− 4.634	–	− 3.220	**1.400**	− 0.297	− 0.086

The three-level grammar S10M20F shares the same behaviour as S10M until generation 20 with an inflection point at generation 10 and a small improvement of the fitness after the introduction of M. Introducing F at generation 20 compensates for the weakness of M. However, given the time/computation wasted with M, S10M20F is not able to catch up with the performance of S10F. While S10M20F does not achieve the best results, it shows little sign of convergence and seems likely to continue its improvement with more generations.

Looking at the individual grammar configurations (i.e., S, M and F), we see that S achieves the worst results, M achieves average results and only F allows achieving a good performance. This clearly indicates that most terminals in F that are not in M are necessary for a better performance despite not being the most important for the practitioners. Therefore, as a rule of thumb, we can say that having the full grammar at the end of the grammar configuration is mandatory for achieving a good performance—without guaranteeing a good performance though (we do not achieve the best results with the grammar configuration S10M20F). This also applies to M and S: given that M converges to better individuals than S, we can say that most terminals in M that are not in S are necessary for a better performance.

Table 3 shows the mean, min, max and standard deviation over 100 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 and Effect Size

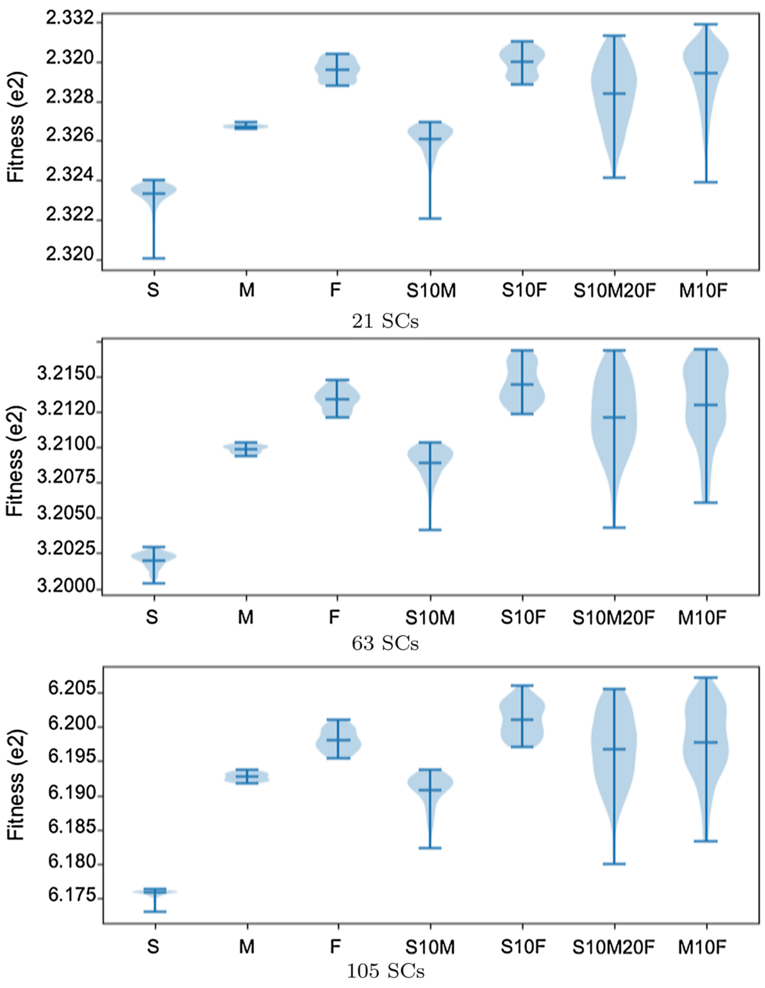


Fig. 5 Violin plot of the best fitness achieved by G3P with the various grammar configurations over the different 100 runs

when comparing every approach against G3P with the grammar configuration F. All G3P algorithms which run multi-level grammar configurations use the population porting strategy when updating the grammar.

Table 3 confirms results that were observed in Fig. 4 as it shows that using G3P with the grammar configuration S10F achieves the best mean results in all instances. In addition, Table 3 consolidates these results as it demonstrates that G3P achieves statistically better results when using S10F rather than F alone, with an Effect Size that is always larger than 0.5 (medium on 21 SCs, and large on 63 and 105 SCs).

Unlike the mean results, G3P with M10F achieves the best max fitness function in all instances. Furthermore, looking at Fig. 5 which show a violin plot of best fitness functions obtained by G3P with each grammar configuration over the 100 runs, we see a non-marginal density of solutions with M10F close to the max result with that same grammar configuration. This seems to indicate that the fact that M10F achieves the maximum best fitness over the 10 runs is not the effect of an outlier run. Therefore, in a situation where it is allowed multiple runs, the user might favour M10F as M10F managed to find the individual with the best fitness in all scenarios when run 100 times. Regarding the min results, we do not see a clear winner. G3P achieves two times out of three the best min results using S10F, while achieving one time out of three the best min results using F. Despite not being as important as mean results, min results can be important as a measure to lower regret when the user is only allowed one run and wants to have a 'guarantee' on the minimum performance.

When it comes to the standard deviation, we see rather high values with respect to the difference in mean values. We notice a tendency for S and M to have the lowest standard deviation which could be explained by the smaller search space due to the restricted grammars. We also see an average standard deviation in case of F, S10M and S10F. The largest standard deviations are observed with S10M20F and M10F which we believe is due to the lack of convergence in their populations.

As a summary, we can say that there is a multi-level grammar that is beneficial in most scenarios. In particular, the two-level grammar which starts with the small grammar before introducing the full one at generation 10 is the grammar configuration that outperforms the use of a single grammar in our case.

6.2 Sensitivity of the approach to the introduction phase

After showing that G3P with the multi-level grammar approach can statistically outperform G3P with a single grammar, using the two-level grammar S10F particularly showed significantly better mean results than F in all instances. We seek in this section to analyse the degree to which the results are sensitive to the introduction phase of the full grammar. In other words, we study how do results vary when modifying the generation at which we change from the small to the full grammar.

To this end, we design 6 configurations SxF with $x \in \{1, 5, 10, 20, 30 \text{ and } 50\}$ which all start with the small grammar and introduce the full grammar at a given generation while porting the entire population. We, therefore, compare G3P on the different instances (i.e., 21, 63 and 105 SCs) using the following grammar configurations: S1F, S5F, S10F, S20F, S30F and S50F. Note that while we choose to study the sensitivity of SxF, the same process could have been performed on MxF (M10F achieves best max results). Choosing SxF is based on the fact that S10F achieves statistically best mean results, which 'ensures' a good quality of results with a lesser regret for its users.

Figure 6 shows the evolution of the best fitness obtained by G3P on the different instances (i.e., 21, 63 and 105 SCs) starting with the small grammar S and introducing the full one F at various generations (i.e., 1, 5, 10, 20, 30 and 50). Results are averaged over 100 runs. G3P uses the population porting strategy when changing the grammar from the small to the full one.

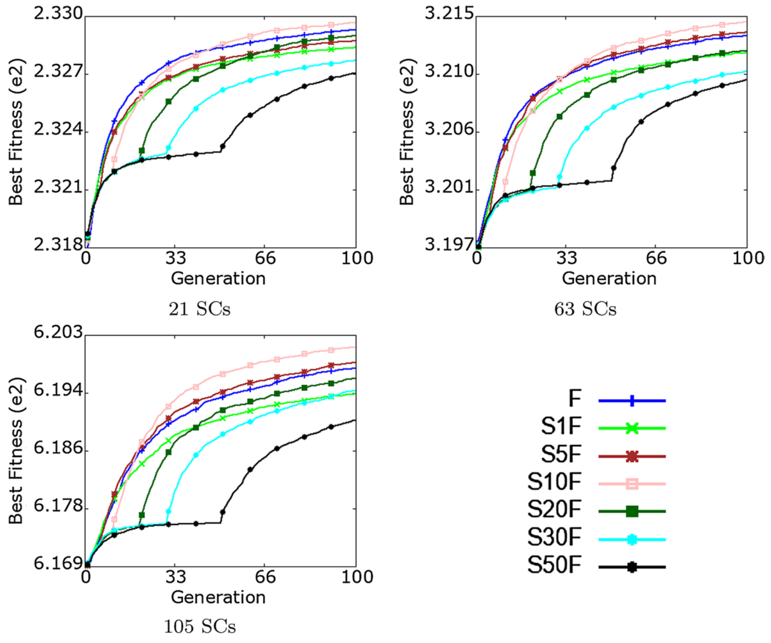


Fig. 6 Mean over 100 runs of the evolution of the best fitness obtained by G3P on the different instances (i.e., 21, 63 and 105 SCs) starting with the small grammar S and introducing the full one F at various generations (i.e., 1, 5, 10, 20, 30 and 50) with the population porting strategy

We see in Fig. 6 a similar trend as in Fig. 4 whereby the G3P algorithm improves the best fitness function in all instances with all the grammar introduction phases it is applied with. We also see that limiting the number of generations to 100 does not seem to allow to achieve a full convergence for the various algorithms and increasing this parameter is likely to yield a better overall performance. We notice that S10F is the only multi-level grammar configuration which clearly outperforms F. The second best result is obtained when introducing F at generation 5 (i.e., S5F) and it already has mixed results in comparison to F: while G3P achieves slightly better results with S5F than with F on 63 and 105 SCs, G3P achieves worse results on 21 SCs with S5F than with F.

We clearly see that the rest of the introduction phases (i.e., 1, 20, 30 and 50) are performing poorly. Introducing the full grammar after generation 1 acts mostly as a handicap where the initiation population is generated using a limited number of terminals and is not evolved using that grammar to find interesting individuals (thus losing the advantage of using a multi-level grammar approach). When it comes to introduction phases strictly larger than 10 (i.e., 20, 30 and 50), we see a clear trend whereby the more the introduction of the full grammar is delayed the worse are the results. This could be explained by either: (1) the introduction phase is too late to allow G3P to achieve good results or (2) the populations that result from the small grammar are too converged leading G3P to waste too much computation to introduce diversity after introducing the full grammar.

Table 4 Mean, min, max and standard deviation (Sd) over 100 runs of the best fitness obtained by G3P when starting with S and introducing F at different generations (i.e., 1, 5, 10, 20, 30 and 50) with a population porting strategy. We also include the significance test *p* value (Mann–Whitney U test with Bonferroni adjustment) and the effect size for each grammar configuration against F. We put in bold the best mean, min, max, significant *p* values, and positive medium/large effect sizes (i.e., larger than 0.5)

Instance	Metric	F	S1F	S5F	S10F	S20F	S30F	S50F
21 SCs	Mean	232.957	232.862	232.898	**232.995**	232.925	232.793	232.726
	Min	**232.880**	232.700	232.756	232.868	232.811	232.685	232.598
	Max	233.040	232.977	233.010	**233.102**	233.037	232.912	232.845
	Sd	0.051	0.077	0.072	0.063	0.067	0.068	0.079
	p value	–	**1.52E−08**	**2.33E−04**	**3.10E−03**	**4.44E−02**	**5.62E−16**	**2.47E−17**
	Effect size	–	− 1.434	− 0.937	**0.647**	− 0.539	− 2.687	− 3.455
63 SCs	Mean	321.334	321.202	321.364	**321.443**	321.216	321.051	320.984
	Min	321.206	320.941	321.169	**321.238**	321.029	320.799	320.719
	Max	321.467	321.379	321.556	**321.681**	321.445	321.208	321.165
	Sd	0.075	0.123	0.117	0.131	0.121	0.124	0.120
	p value	–	**2.32E−07**	8.20E−01	**4.73E−05**	**1.12E−05**	**3.55E−17**	**2.47E−17**
	Effect size	–	− 1.281	0.302	**0.971**	− 1.158	− 2.726	− 3.457
105 SCs	Mean	619.799	619.426	619.880	**620.101**	619.650	619.473	619.047
	Min	619.537	618.967	619.514	**619.708**	619.246	619.009	617.882
	Max	620.107	619.833	620.380	**620.598**	620.040	619.889	619.522
	Sd	0.160	0.234	0.263	0.244	0.214	0.253	0.367
	p value	–	**2.60E−11**	8.99E−01	**1.93E−09**	**1.44E−03**	**1.52E−08**	**2.47E−17**
	Effect size	–	− 1.842	0.366	**1.400**	− 0.780	− 1.524	− 2.626

This is what we will study below by analysing what happens to the populations when introducing the full grammar at different generations.

Before moving to the population analysis, we report in Table 4 the mean, min, max and standard deviation of the performance achieved by G3P when starting with the small grammar and introducing the full one at various generations (i.e., 1, 5, 10, 20, 30 and 50). We also include the *p* value and Effect Size when comparing every approach against G3P with the grammar configuration F.

Table 4 confirms what we noticed in Fig. 6 and shows that S1F, S20F, S30F and S50F are significantly worse than F in all instances, with Effect Sizes that are either medium (2 out of 12 cases) or large (10 out of 12 cases). It also shows that in addition to S5F being significantly worse than F on 21 SCs with a large Effect Size, it is not even statistically better on 105 SCs. Therefore, an analysis of what makes the two-level approach with the population porting strategy so sensitive to the phase at which the full grammar is introduced is needed.

6.3 Analysing reasons of the sensitivity to the introduction phase

We would like to analyse what makes G3P with the two-level grammar using the population porting strategy so sensitive to the phase at which the full grammar is introduced. Therefore, we plot the fitness function of each individual in the population at each generation from the run with the median best fitness and study both their performance and diversity.

Figure 7 shows the fitness of every individual at each generation extracted from the run with the median best fitness, with G3P starting by the small grammar and introducing the full grammar at various introduction phases (i.e., 1, 5, 10, 20, 30 and 50).

We see in the top row of Fig. 7 the fitness of individuals when starting G3P with the small grammar and introducing the full one at the first generation without allowing much evolution with the small grammar. We notice that the fitness of the initial population is relatively well-spread. However, we see a period of fast convergence of the population following the introduction of the full grammar (until generation 20, 10 and 10 for instances 21, 63 and 105 SCs respectively). We clearly see that G3P loses in the diversity of the population during that period and uses several generations to bring it to a level that it will then keep until the end of the evolution. We also see that the algorithm has a marginal improvement of the best performance during the period of low diversity, while the performance spikes when the diversity is brought to the desired level.

We see in the second and third rows of Fig. 7 the fitness of individuals when starting G3P with the small grammar and introducing the full one at the generations 5 and 10 respectively. We see that with the exception of S5F on 105 SCs, the populations do not converge too much and keep a decent level of diversity allowing G3P to either maintain or easily increase the diversity (e.g., such as in S10F on 21 SCs) to a 'suitable' level.

Figures in row four correspond to the introduction of the full grammar in generation 20. In these figures, we see a regular distribution from the beginning to the end of the evolution. we also see a continual improvement of the best fitness. These two elements let us think that, in this particular context, the late introduction of the full grammar is the main reason for the poor performance of G3P using the S20F grammar configuration and a larger number of generation might improve the results. Another issue that we notice is that the improvement of the best performance does not happen directly after the introduction of the full grammar at generation 20, but in a slightly delayed generation (around generation 25). This exhibits a weakness of the population porting strategy to introduce new terminals from the new grammar in a fast pace.

The last two rows of Fig. 7 correspond to the introduction of the full grammar at generations 30 and 50 respectively. We see a similar behaviour as in row four (introducing the full grammar at generation 20) as the improvement of the best fitness does not appear directly after the introduction of the full grammar. This is mainly due to the time it takes for G3P to introduce new terminals from the full grammar to the population. This is also due, in some cases, to the lack of diversity in the population (e.g., S30F and S50F on 105 SCs) that is exacerbated by multiple generations using the small grammar.

Figure 8 shows the evolution of the diversity (measured as the distance between the best fitness and average fitness) in the population of every generation of G3P

Fig. 7 Fitness of every individual at each generation from the median run, with G3P starting with the small grammar and introducing the full grammar at various introduction phases (i.e., 1, 5, 10, 20, 30 and 50) using the population porting strategy ▶

with the various grammar configurations on the different instances (i.e., 21, 63 and 105 SCs). Results are averaged over 100 runs.

We see in Fig. 8 a similar trend as with Fig. 7. The initial populations that are generated with each grammar configuration are of a high diversity. Although, the initial populations generated with the grammar S are slightly more diverse than those with the grammar F. We see that, with all the grammar configurations, the diversity drops drastically after the first generation of G3P. However, the diversity with the grammar configuration F seems to stabilise at a descent level afterwards. In the contrary, the diversity with the two-level grammars drop further than with the grammar configuration F, before increasing to reach its level in later generations.

We notice three trends with the two-level grammar configurations. The first trend is with S1F where the diversity drop at the first generation and increases directly afterwards. However, it takes G3P with S1F several generations to level the diversity with the grammar configuration F. The second trend is with the grammar configurations S5F, S10F and S20F where the diversity drops even further after few generations (from generation 1 to the generation at which the grammar F is introduced). However, the diversity with these grammar configurations levels the diversity with the grammar configuration F quickly (in few generations after the introduction of the grammar F). The last trend we see is with the grammar configurations S30F and S50F. After a diversity drop at the first generation, this drop continues even further until the introduction of the grammar F. Unlike with S5F, S10F and S20F, after the introduction of grammar F, the diversity takes longer to reach the level of diversity with the grammar configuration F.

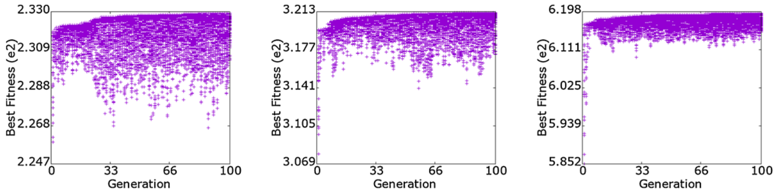
As a summary, we can say that starting with S and introducing F at a later generation with population porting is highly sensitive to the generation at which the grammar is updated. When introducing the full grammar at early generations, we take the risk of lacking diversity in the population. Whereas, when introducing the full grammar at a later generation, we take the risk of lack of execution budget, lack of diversity / convergence of the population, and slow introduction of new terminals from the full grammar into the population.

7 Multi-level grammar through seeding

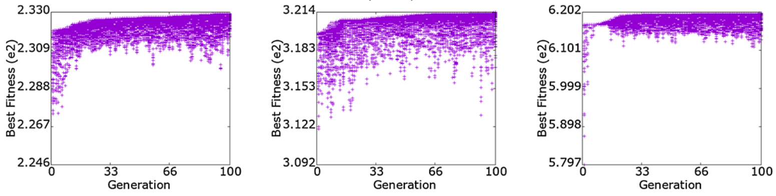
In the previous multi-level grammar approach, we were 'porting' the entire population when performing the grammar update. We saw that it leads to an undesirable evolutionary behaviour. The population that follows the introduction of the new grammar either lacks diversity or spends many generations to introduce new terminals instead of using this computational power to improve the performance.

In this section, we replace the population porting strategy with the seeding strategy, whereby only the best individual obtained from the small grammar is kept, and

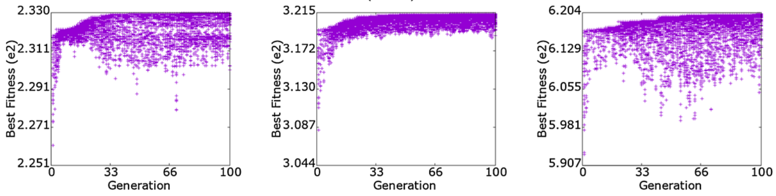
21 SCs 63 SCs 105 SCs



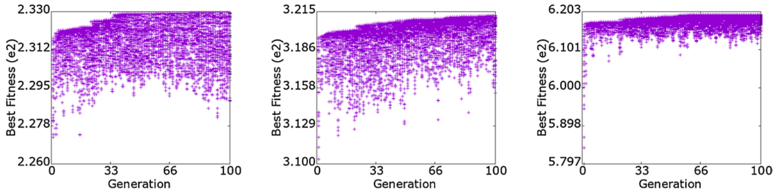
(S1F)



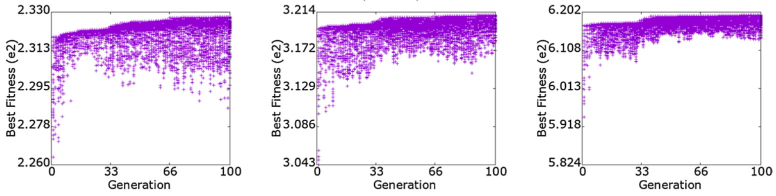
(S5F)



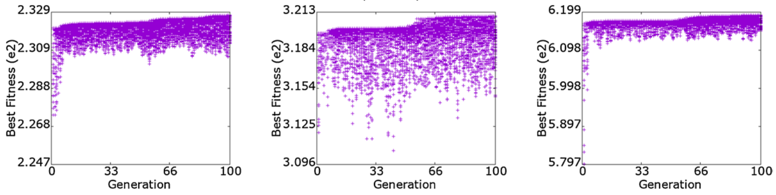
(S10F)



(S20F)



(S30F)



(S50F)

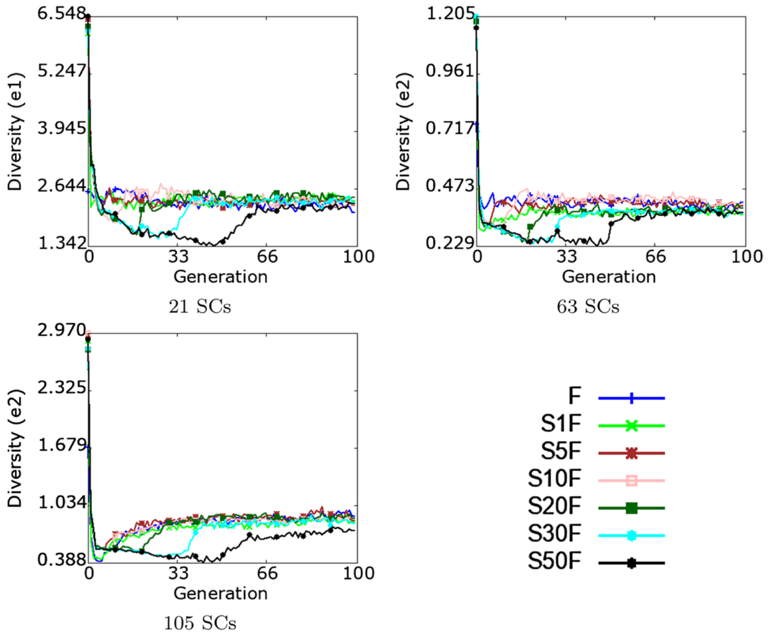


Fig. 8 The evolution of the diversity (measured as the distance between the best fitness and average fitness) in the population of every generation of G3P with the various grammar configurations on the different instances (i.e., 21, 63 and 105 SCs). Results are averaged over 100 runs

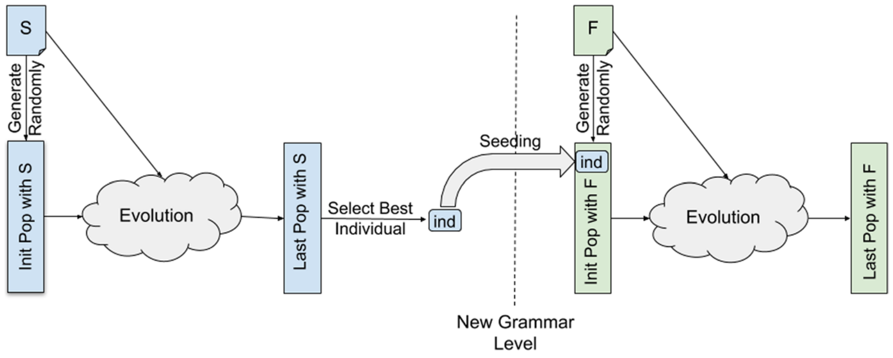


Fig. 9 Example of the multi-level grammar approach with the seeding strategy. G3P generates individuals using the grammar S and evolves them for a few generations using that same grammar. Next, G3P takes the best individual that resulted from the evolution with S and generates the remaining individuals to fill a population using the grammar F. The G3P evolves this population using the grammar F for the remaining generations

the rest of the population is generated randomly using the Ramped Half-Half technique on the full grammar. The seeding approach makes the multi-level approach act in a similar way as the transfer learning approach. However, the transfer is within the same evolutionary process (in the same evolutionary run) and corresponds to a knowledge that is learned on the same problem. The seeding approach could also be seen as an extinction event that would enhance the diversity in the search process [53] while keeping the elite (though we only keep one elite individual in this work).

Figure 9 shows an example of the multi-level grammar approach where the population porting strategy is replaced with the seeding one. Unlike in the population porting, the population that resulted of the evolution with the small grammar S is not used as an initial population to the evolution with the full grammar F. Instead, only the best individual from S is selected and injected (seeded) to a randomly generated initial population with F.

We expect that the randomly generated individuals using the full grammar will bring the diversity to an adequate level, while at the same time, introducing individuals with terminals from the new grammar immediately after the grammar modification. Therefore, the seeding strategy seems to cope with the shortcomings of the population porting strategy that was used earlier. We anticipate that the new randomly generated individuals to be of a rather poor quality in terms of fitness. However, we believe that the individual that will be kept from the previous grammar as a seed would act as an attractor and a driver for the improvement of the new population. Additionally, randomly regenerating an entire population has a computational cost. This process consumes an entire generation (minus one individual). However, we believe that wasting one generation would still have a smaller impact on the evolution than the numerous generations wasted by the population porting strategy.

7.1 Comparing the seeding and the population porting strategies

Figure 10 shows the evolution of the best fitness over generations on the different instances (i.e., 21, 63 and 105 SCs) with G3P starting with the small grammar and introducing the full one at different phases (i.e., generations 1, 5, 10, 20, 30 and 50). Results are averaged over 100 runs. In each of the 18 cases in Fig. 10 which combine the grammar configuration and the instance, we plot the results obtained when using either the population porting or the seeding strategy. We also plot results obtained using the full grammar for reference.

The first thing we notice from Fig. 10 is that similarly to the population porting strategy, G3P successfully improves the performance when using the seeding strategy with all the grammar configurations and in all instances.

We also see that while the seeding strategy starts with the same performance as the population porting one from the initial generation until the generation at which the full grammar is introduced, the seeding achieves a better performance than the population porting in all the 18 studied cases (grammar configuration by instance). This is a clear indication that the seeding strategy is the element which leads to this improvement. The seeding strategy achieves a slight improvement over the population porting strategy when using the grammar configurations S5F and S10F. However, the improvement is larger when it comes to other grammar configurations. It also seems to be more important the later the full grammar is introduced. In other

Fig. 10 Mean over 100 runs of the evolution of the best fitness obtained by G3P using both the population porting and the seeding strategies on the different instances (i.e., 21, 63 and 105 SCs) starting with the small grammar S and introducing the full one F at various generations (i.e., 1, 5, 10, 20, 30 and 50). Results obtained with G3P using the full grammar configuration are also shown for comparison ▶

words, seeding outperforms population porting more on S30F than on S20F, and more on S50F than on S30F.

When comparing results obtained using the seeding strategy to those obtained with full grammar configuration, we see that with the exception of two cases with S50F (i.e., 21 and 63 SCs), seeding achieves either similar or better results in all the other 16 cases. Among these 16 cases, seeding achieves similar results than the full grammar configuration on 7 cases, while significantly outperforming it on the 9 other cases. This is a significant improvement from the population porting strategy which only achieves similar or better results than the full grammar configuration in 7 cases (i.e., 5 similar and 2 better results), and worse results on 11 cases.

We see that the seeding is an efficient way to improve results of the two-level grammar especially on S1F, S20F and S30 where the population porting strategy achieved worse results than the full grammar, whereas using the seeding strategy allows those two-level grammars to equalise or outperform the full one in all instances. We see that in the two latter grammar configurations (i.e., S20F and S30F), the seeding improves on the full grammar, whereas in S1F, it only equalises it as the initial population is regenerated at generation one without seeding any interesting individual.

After examining the evolution of the population for the median run of G3P with the seeding approach when starting with S and introducing F at the aforementioned phases, we notice that the seeding approach quickly introduces a desired diversity to the population after the grammar update. At the same time, the seeding strategy shortens the time wasted by the approach as it starts improving the best fitness only a few generations after introducing the new grammar, which indicates a fast introduction of the new terminals.

Table 5 shows the mean, min, max and standard deviation obtained by G3P using either the grammar configuration F or by starting with the small grammar and introducing the full one at various generations (i.e., 1, 5, 10, 20, 30 and 50) with both population porting and seeding strategies on the different instances (i.e., 21, 63 and 105 SCs). Table 5 also includes the *p* value significance test (computed using the Mann–Whitney U test with Bonferroni adjustment) and the substantive significance (using Cohen's d Effect Size) between the seeding strategy and either of the two other techniques (i.e., F or population porting) in each of the 18 scenarios (grammar configuration by instance). It shows in bold the best result between the different strategies in each scenario in addition to the significant *p* values and positive medium/large Effect Sizes (i.e., larger than 0.5).

Table 5 confirms the supremacy of the seeding strategy over the population porting one as it shows that the seeding strategy achieves statistically better mean results than the population porting strategy in 12 out of 18 scenarios (only 6 cases out of 18 where results are not statistically significant), with an effect that is medium (3

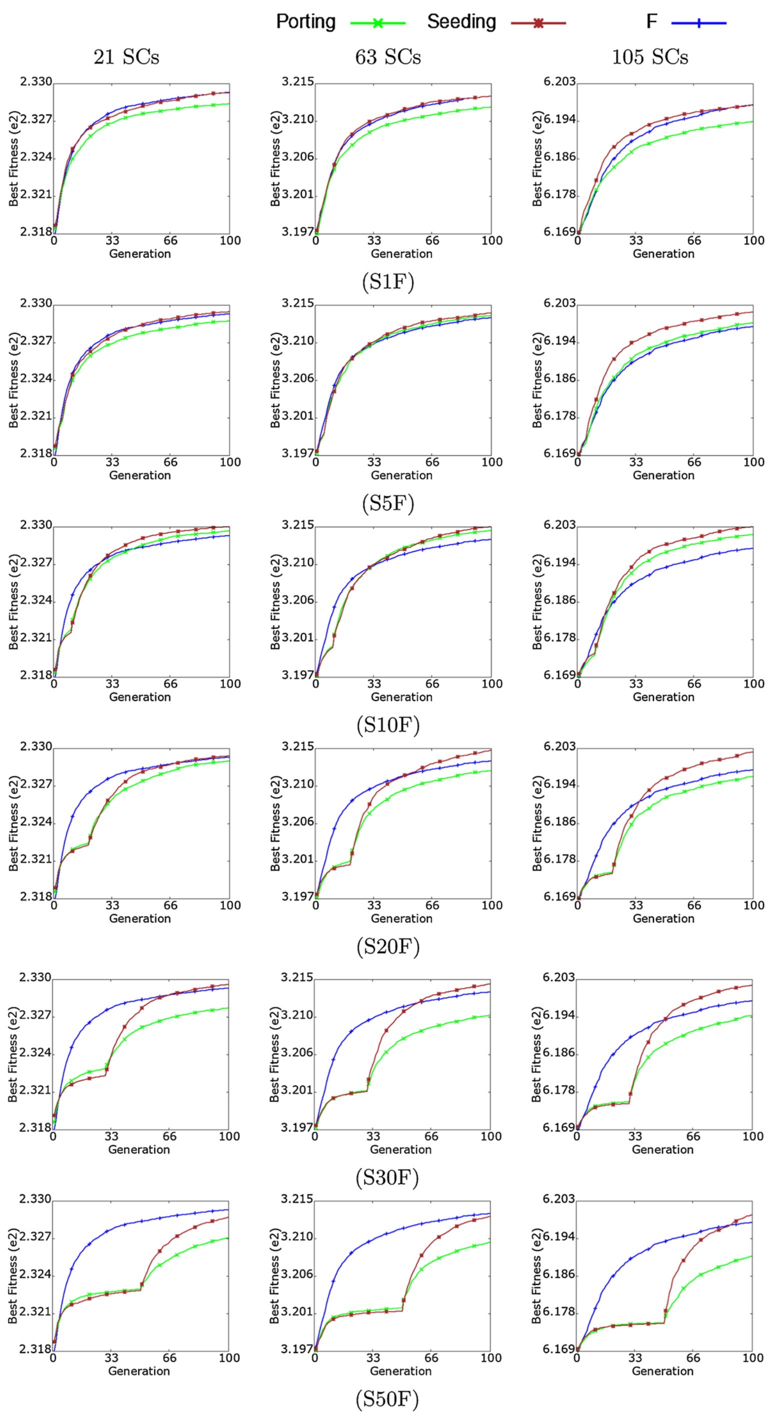

Porting
Seeding
F
21 SCs
63 SCs
105 SCs
Best Fitness (e2)
Generation
0
33
66
100
2.318
2.321
2.324
2.327
2.330
3.197
3.201
3.206
3.210
3.215
6.169
6.178
6.186
6.194
6.203
(S1F)
(S5F)
(S10F)
(S20F)
(S30F)
(S50F)

Table 5 Mean, min, max and standard deviation (Sd) achieved by G3P with F or starting with the small grammar and introducing the full one at various generations (i.e., 1, 5, 10, 20, 30 and 50) with both population porting and seeding strategies on the different instances (i.e., 21, 63 and 105 SCs). We also include the *p* value significance test (Mann–Whitney U test with Bonferroni adjustment) and Effect Size between the population porting and seeding strategies in each scenario. In bold are best results between F, seeding and population porting, in addition to significant *p* values and positive medium/large effect sizes (i.e., larger than 0.5) between seeding and the other strategies

Grammar	Function	21SCs			63SCs			105SCs		
		F	Porting	Seeding	F	Porting	Seeding	F	Porting	Seeding
S1F	Mean	**232.957**	232.862	232.954	**321.334**	321.202	**321.334**	619.799	619.426	**619.801**
	Min	**232.880**	232.700	232.837	**321.206**	320.941	321.076	**619.537**	618.967	619.287
	Max	233.040	232.977	**233.055**	321.467	321.379	**321.586**	620.107	619.833	**620.373**
	Sd	0.051	0.077	0.065	0.075	0.123	0.160	0.160	0.234	0.324
	p value	1.00E+00	**4.40E−07**	–	**1.81E−03**	1.00E+00	–	**1.84E−07**	1.00E+00	–
	Effect size	− 0.051	**1.272**	–	0.001	**0.915**	–	0.006	**1.313**	–
S5F	Mean	232.957	232.898	**232.975**	321.334	321.364	**321.394**	619.799	619.880	**620.119**
	Min	**232.880**	232.756	232.874	**321.206**	321.169	321.176	619.537	619.514	**619.644**
	Max	233.040	233.010	**233.071**	321.467	321.556	**321.604**	620.107	620.380	**620.464**
	Sd	0.051	0.072	0.060	0.075	0.117	0.134	0.160	0.263	0.268
	p value	4.50E−01	**5.42E−06**	–	7.03E−01	1.43E−01	–	**6.61E−05**	**1.64E−07**	–
	Effect size	0.314	**1.142**	–	**0.548**	0.238	–	**1.432**	**0.890**	–
S10F	Mean	232.957	232.995	**233.029**	321.334	321.443	**321.489**	619.799	620.101	**620.275**
	Min	232.880	232.868	**232.931**	321.206	**321.238**	321.227	619.537	619.708	**619.839**
	Max	233.040	233.102	**233.116**	321.467	**321.681**	321.674	620.107	620.598	**620.652**
	Sd	0.051	0.063	0.051	0.075	0.131	0.144	0.160	0.244	0.226
	p value	**4.08E−08**	**1.42E−02**	–	**2.82E−02**	**8.97E−07**	–	**3.84E−04**	**9.28E−14**	–
	Effect size	**1.406**	**0.586**	–	**1.335**	0.333	–	**2.400**	**0.729**	–

Table 5 (continued)

Grammar	Function	21SCs			63SCs			105SCs		
		F	Porting	Seeding	F	Porting	Seeding	F	Porting	Seeding
S20F	Mean	232.957	232.925	**232.968**	321.334	321.216	**321.463**	619.799	619.650	**620.194**
	Min	**232.880**	232.811	232.824	321.206	321.029	**321.210**	619.537	619.246	**619.645**
	Max	233.040	233.037	**233.107**	321.467	321.445	**321.695**	620.107	620.040	**620.560**
	Sd	0.051	0.067	0.093	0.075	0.121	0.147	0.160	0.214	0.268
	p value	1.00E+00	7.59E−02	–	**1.35E−10**	**3.10E−05**	–	**4.51E−13**	**3.76E−10**	–
	Effect size	0.146	**0.532**	–	**1.100**	**1.823**	–	**1.768**	**2.218**	–
S30F	Mean	232.957	232.793	**232.987**	321.334	321.051	**321.431**	619.799	619.473	**620.143**
	Min	232.880	232.685	**232.884**	321.206	320.799	**321.216**	619.537	619.009	**619.690**
	Max	233.040	232.912	**233.086**	321.467	321.208	**321.658**	620.107	619.889	**620.867**
	Sd	0.051	0.068	0.058	0.075	0.124	0.136	0.160	0.253	0.263
	p value	**4.53E−02**	**6.99E−17**	–	**2.12E−17**	**1.77E−03**	–	**1.20E−15**	**2.55E−09**	–
	Effect size	**0.548**	**3.023**	–	**0.873**	**2.890**	–	**1.561**	**2.568**	–
S50F	Mean	**232.957**	232.726	232.895	**321.334**	320.984	321.302	619.799	619.047	**619.963**
	Min	**232.880**	232.598	232.751	**321.206**	320.719	321.051	**619.537**	617.882	619.337
	Max	233.040	232.845	**233.080**	321.467	321.165	**321.644**	620.107	619.522	**620.598**
	Sd	0.051	0.079	0.088	0.075	0.120	0.168	0.160	0.367	0.384
	p value	**1.02E−03**	**2.83E−12**	–	**3.56E−14**	4.09E−01	–	**5.40E−16**	2.82E−01	–
	Effect size	− 0.852	**2.011**	–	− 0.243	**2.157**	–	**0.551**	**2.413**	–

out of 18 cases) and large (13 out of 18 cases). It also shows that the seeding strategy achieves better min and max results than the population porting one in all scenarios. Regarding the standard deviation, we see that both strategies have similar standard deviation values despite being slightly larger for the seeding in 13 out of 18 scenarios.

Table 5 confirms that G3P with the seeding strategy using S10F achieves statistically better mean results with large Effect Sizes over F and the other grammar configurations in all instances. It also shows that S10F achieves better min results than F in all instances. However, S10F with seeding only achieves best max results on instance 21 SCs, whereas this position is taken by S20F and S30F with seeding on 63 and 105 SCs respectively. As the problem complexity increases (more SCs in the instance) it seems better to delay the introduction phase to achieve the best max performance (S10F, S20F and S30F achieve best max results on 21, 63 and 105 SCs). Table 5 also shows that in addition to S10F with seeding being statistically better than F, three other grammars with seeding are better than F, i.e., S5F, S20F and S30F despite not achieving the best overall results. Having four grammar configurations outperforming the use of a single full grammar is another indicator that the seeding strategy is a better alternative to the population porting one. Additionally, using the grammar configuration F does not outperform clearly the two remaining grammar configurations with seeding (i.e., S1F and S50F). F does not achieve statistically better results than S1F with seeding on any of the instances. Furthermore, it is also statistically worse than S50F with seeding on the instance 105 SCs, with a medium Effect Size.

8 Conclusion

While the grammar-guided genetic programming approach has recently proven its capability to evolve efficient transmission schedulers in heterogeneous networks, some works have shown that there is still room for improvement.

We confirmed in this work that using a multi-level grammar approach allows improving the quality of the schedules obtained by the G3P algorithm. Our proposed approach consists of starting the optimisation with a short and more compact grammar containing only the most important terminals in order to direct the search towards 'ideal' individuals. Then, to introduce a full grammar during the evolution to probe a larger part of the search space and improve the performance. However, despite the improvement, we have shown that multi-level grammar approach is sensitive to the phase at which grammars are introduced.

After studying the way the fitness of individuals evolves from a generation to another, we have identified that the lack of diversity and the delayed introduction of new terminals, combined with the late update of the grammar are the major factors for this sensitivity. Therefore, we proposed to substitute the original population porting strategy with a new one (i.e., the seeding strategy). The population porting strategy consists of taking all individuals from the last population that resulted from the evolution using the small grammar and using them as the initial population for the evolution with the larger grammar. Whereas in the seeding strategy, we only take the best individual as a seed to a new randomly generated population using the new grammar. Moving to the seeding strategy alleviated the shortcomings of the population porting strategy and mitigated the sensitivity of the multi-level grammar approach to the introduction phase of the larger grammar. In addition to mitigating this sensitivity, using the seeding strategy outperforms the use of the population porting in all the studied scenarios and thus improves the performance of the multi-level grammar G3P over a regular G3P algorithm.

Our ultimate goal is to create a reactive multi-level approach that is agnostic of the problem domain. To achieve this, we endeavour to answer many questions in our future work: (1) How to automatically design the restricted grammars? (2) How many grammars to use per level? (3) When to introduce the grammars? and (4) How many individuals to seed from a grammar to another?

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

References

1. T. Saber, J. Thorburn, L. Murphy, A. Ventresque, VM reassignment in hybrid clouds for large decentralised companies: a multi-objective challenge. Future Gener. Comput. Syst. **79**, 751–764 (2018)
2. M. Fenton, D. Lynch, S. Kucera, H. Claussen, M. O'Neill, Multilayer optimization of heterogeneous networks using grammatical genetic programming. IEEE Trans. Cybern. **47**, 2938–2950 (2017)
3. V.N.I. Cisco, Global mobile data traffic forecast update, 2016–2021. White paper (2017)
4. Statista: forecast of mobile phone users worldwide (2018). www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/
5. H.E. Ciritoglu, T. Saber, T.S. Buda, J. Murphy, C. Thorpe, Towards a better replica management for hadoop distributed file system, in *BigData Congress* (2018), pp. 104–111
6. A. Tall, Z. Altman, E. Altman, Self organizing strategies for enhanced ICIC (eICIC), in *WiOpt* (2014), pp. 318–325
7. M. Fenton, D. Lynch, D. Fagan, S. Kucera, H. Claussen, M. O'Neill, Towards automation & augmentation of the design of schedulers for cellular communications networks. Evol. Comput. **3**, 1–30 (2018)
8. J.G. Andrews, S. Buzzi, W. Choi, S.V. Hanly, A. Lozano, A.C. Soong, J.C. Zhang, What will 5G be? IEEE J. Sel. Areas Commun. **32**, 1065–1082 (2014)
9. 3GPP: the 3rd generation partnership project. www.3gpp.org
10. D. Lynch, M. Fenton, S. Kucera, H. Claussen, M. O'Neill, Scheduling in heterogeneous networks using grammar-based genetic programming, in *EuroGP* (2016), pp. 83–98
11. T. Saber, D. Fagan, D. Lynch, S. Kucera, Claussen, H., O'Neill, M., Multi-level grammar genetic programming for scheduling in heterogeneous networks, in *EuroGP*, (2018), pp. 118–134
12. C.E. Shannon, Communication in the presence of noise. IRE **37**, 10–21 (1949)
13. A. Weber, O. Stanze, Scheduling strategies for HetNets using eICIC, in *ICC* (2012), pp. 6787–6791
14. A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs. Eur. J. Oper. Res. **246**, 345–378 (2015)
15. D. Jakobović, K. Marasović, Evolving priority scheduling heuristics with genetic programming. Appl. Soft Comput. **12**, 2781–2789 (2012)
16. J. Branke, S. Nguyen, C.W. Pickardt, M. Zhang, Automated design of production scheduling heuristics: a review. IEEE Trans. Evol. Comput. **20**, 110–124 (2016)
17. S. Nguyen, Y. Mei, M. Zhang, Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. **3**, 41–66 (2017)

18. J. Pang, J. Wang, D. Wang, G. Shen, Q. Jiang, J. Liu, Optimized time-domain resource partitioning for enhanced inter-cell interference coordination in heterogeneous networks, in *WCNC* (2012), pp. 1613–1617
19. D. López-Pérez, H. Claussen, Duty cycles and load balancing in HetNets with eICIC almost blank subframes, in *PIMRC Workshops* (2013), pp. 173–178
20. L. Jiang, M. Lei, Resource allocation for eICIC scheme in heterogeneous networks, in *PIMRC* (2012), pp. 448–453
21. S. Deb, P. Monogioudis, J. Miernik, J.P. Seymour, Algorithms for enhanced inter-cell interference coordination (eICIC) in LTE HetNets. IEEE/ACM Trans. Netw. **22**, 137–150 (2014)
22. D. Fagan, M. Fenton, D. Lynch, S. Kucera, H. Claussen, M. O'Neill, Deep learning through evolution: a hybrid approach to scheduling in a dynamic environment, in *IJCNN* (2017), pp. 775–782
23. L.T. Ho, I. Ashraf, H. Claussen, Evolving femtocell coverage optimization algorithms using genetic programming, in *PIMRC* (2009), pp. 2132–2136
24. E. Hemberg, L. Ho, M. O'Neill, H. Claussen, A comparison of grammatical genetic programming grammars for controlling femtocell network coverage. Genet. Program. Evolvable Mach. **14**, 65–93 (2013)
25. I. Dempsey, M. O'Neill, A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments*, vol. 194 (Springer, New York, 2009)
26. D. Lynch, M. Fenton, S. Kucera, H. Claussen, M. O'Neill, Evolutionary learning of scheduling heuristics for heterogeneous wireless communications networks, in *GECCO* (2016), pp. 949–956
27. T. Saber, D. Fagan, D. Lynch, S. Kucera, H. Claussen, M. O'Neill, A hierarchical approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks, in *TPNC* (2018)
28. T. Saber, A. Ventresque, X. Gandibleux, L. Murphy, GeNePi: a multi-objective machine reassignment algorithm for data centres, in *HM* (2014), pp. 115–129
29. T. Saber, F. Delavernhe, M. Papadakis, M. O'Neill, A. Ventresque, A hybrid algorithm for multi-objective test case selection, in *CEC* (2018)
30. T. Saber, A. Ventresque, J. Marques-Silva, J. Thorburn, L. Murphy, MILP for the multi-objective VM reassignment problem, in *ICTAI* (2015), pp. 41–48
31. T. Saber, J. Marques-Silva, J. Thorburn, A. Ventresque, Exact and hybrid solutions for the multi-objective VM reassignment problem. Int. J. Artif. Intell. Tools **26**, 1760004 (2017)
32. R.I.B. McKay, T.H. Hoang, D.L. Essam, X.H. Nguyen, Developmental evaluation in genetic programming: the preliminary results, in *EuroGP* (2006), pp. 280–289
33. T.H. Hoang, R.I. McKay, D. Essam, X.H. Nguyen, Developmental evaluation in genetic programming: a position paper, in *FBIT* (2007), pp. 773–778
34. T.H. Hoang, D. Essam, N.X. Hoai et al., Developmental evaluation in genetic programming: the tag-based frame work. Int. J. Knowl. Intell. Eng. Syst. **12**, 69–82 (2008)
35. T.H. Hoang, D. Essam, B. McKay, N.X. Hoai, Building on success in genetic programming: adaptive variation and developmental evaluation, in *ISCIA* (2007), pp. 137–146
36. N.F. McPhee, E. Crane, S.E. Lahr, R. Poli, Developmental plasticity in linear genetic programming, in *GECCO* (2009), pp. 1019–1026
37. T.H. Hoang, R.I. McKay, D. Essam, N.X. Hoai, On synergistic interactions between evolution, development and layered learning. IEEE Trans. Evol. Comput. **15**, 287–312 (2011)
38. S.M. Gustafson, W.H. Hsu, Layered learning in genetic programming for a cooperative robot soccer problem, in *EuroGP* (2001), pp. 291–301
39. D. Jackson, A.P. Gibbons, Layered learning in Boolean GP problems, in *EuroGP* (2007), pp. 148–159
40. N.T. Hien, N.X. Hoai, B. McKay, A study on genetic programming with layered learning and incremental sampling, in *CEC* (2011), pp. 1179–1185
41. T.H. Nguyen, X.H. Nguyen, Learning in stages: a layered learning approach for genetic programming, in *RIVF* (2012), pp. 1–4
42. T.T.H. Dinh, T.H. Chu, Q.U. Nguyen, Transfer learning in genetic programming, in *CEC* (2015), pp. 1145–1151
43. E. Haslam, B. Xue, M. Zhang, Further investigation on genetic programming with transfer learning for symbolic regression, in *CEC* (2016), pp. 3598–3605
44. M. Iqbal, B. Xue, H. Al-Sahaf, M. Zhang, Cross-domain reuse of extracted knowledge in genetic programming for image classification. IEEE Trans. Evol. Comput. **21**, 569–587 (2017)

45. R.I. Mckay, N.X. Hoai, P.A. Whigham, Y. Shan, M. O'neill, Grammar-based genetic programming: a survey. Genet. Program. Evolvable Mach. **11**, 365–396 (2010)
46. M. Fenton, J. McDermott, D. Fagan, S. Forstenlechner, E. Hemberg, M. O'Neill, PonyGE2: grammatical evolution in python, in *GECCO* (2017), pp. 1194–1201
47. T. Saber, D. Brevet, G. Botterweck, A. Ventresque, Is seeding a good strategy in multi-objective feature selection when feature models evolve? Inf. Softw. Technol. **61**, 33–51 (2017)
48. C. Ryan, R.M.A. Azad, Sensible initialisation in grammatical evolution, in *GECCO* (2003), pp. 142–145
49. P. Royston, Approximating the shapiro-wilk W-test for non-normality. Stat. Comput. **2**, 117–119 (1992)
50. G.D. Garson, *Testing Statistical Assumptions* (Statistical Associates Publishing, Asheboro, 2012)
51. A. Arcuri, L. Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, in *ICSE* (2011), pp. 1–10
52. G.M. Sullivan, R. Feinn, Using effect size—or why the p value is not enough. J. Grad. Med. Educ. **4**, 279–282 (2012)
53. J. Lehman, R. Miikkulainen, Enhancing divergent search through extinction events, in *GECCO* (2015), pp. 951–958

Affiliations

Takfarinas Saber[1] · David Fagan[2] · David Lynch[2] · Stepan Kucera[3] · Holger Claussen[3] · Michael O'Neill[2]

David Fagan
david.fagan@ucd.ie

David Lynch
david.lynch@ucd.ie

Stepan Kucera
stepan.kucera@nokia-bell-labs.com

Holger Claussen
holger.claussen@nokia-bell-labs.com

Michael O'Neill
m.oneill@ucd.ie

[1] School of Computer Science, University College Dublin, Dublin, Ireland

[2] Natural Computing Research and Applications Group, School of Business, University College Dublin, Dublin, Ireland

[3] Bell Laboratories, Nokia, Dublin, Ireland