Genet Program Evolvable Mach (2013) 14:65–93
DOI 10.1007/s10710-012-9171-8

A comparison of grammatical genetic programming grammars for controlling femtocell network coverage

Erik Hemberg · Lester Ho · Michael O'Neill · Holger Claussen

Received: 29 September 2011 / Revised: 18 September 2012 / Published online: 6 October 2012

Abstract We study grammars used in grammatical genetic programming (GP) which create algorithms that control the base station pilot power in a femtocell network. The overall goal of evolving algorithms for femtocells is to create a continuous online evolution of the femtocell pilot power control algorithm in order to optimize their coverage. We compare the performance of different grammars and analyse the femtocell simulation model using the grammatical genetic programming method called grammatical evolution. The grammars consist of conditional statements or mathematical functions as are used in symbolic regression applications of GP, as well as a hybrid containing both kinds of statements. To benchmark and gain further information about our femtocell network simulation model we also perform random sampling and limited enumeration of femtocell pilot power settings. The symbolic regression based grammars require the most configuration of the evolutionary algorithm and more fitness evaluations, whereas the conditional statement grammar requires more domain knowledge to set the parameters. The content of the resulting femtocell algorithms shows that the evolutionary computation (EC) methods are exploiting the assumptions in the model. The ability of EC to exploit bias in both the fitness function and the

Recommended by Una-May O'Reilly and Steven Gustafson.

E. Hemberg (✉) · M. O'Neill
Complex and Adaptive Systems Laboratory, School of Computer Science and Informatics,
University College Dublin, Dublin, Ireland
e-mail: erik.hemberg@ucd.ie

M. O'Neill
e-mail: m.oneill@ucd.ie

L. Ho · H. Claussen
Bell Laboratories, Alcatel-Lucent, Dublin, Ireland
e-mail: lester.ho@alcatel-lucent.com

H. Claussen
e-mail: holger.claussen@alcatel-lucent.com

underlying model is vital for identifying the current system and improves the model and the EC method. Finally, the results show that the best fitness and engineering performances for the grammars are similar over both test and training scenarios. In addition, the evolved solutions' performance is superior to those designed by humans.

Keywords Genetic programming · Grammars · Femtocell · Symbolic regression · Grammatical evolution

1 Introduction

In telecommunication networks, femtocells are low power, low-cost, user-deployed cellular base stations (BSs) with a typical coverage range of 10s of meters [4]. Femtocells are currently on sale worldwide to consumers and enterprises. In order to minimise operational expenses, femtocells have self-configuration and self-optimisation capability to enable plug-and-play deployment. These capabilities are implemented using algorithms that are designed to automatically change certain network configuration parameters in response to any changes in the network environment. Additionally, to maintain scalability when used in large networks, these algorithms should work in a distributed manner whenever possible, using only local information but still achieving good global performance.

Designing these highly distributed algorithms can be difficult, especially if the network environment varies significantly and multiple conflicting objectives exist. In addition, for femtocell deployments in enterprise environments, a group of femtocells is deployed where the individual cells need to work together to jointly provide continuous coverage in a large building or outdoor area. In Fig. 1 the coverage of a femtocell setup for an office environment with 12 femtocell BSs is shown, with colour indicating areas of femtocell coverage—the BSs are using an evolved coverage algorithm. When femtocell users enter any gaps (white) in the coverage between the femtocells, mobility procedures (handovers or cell re-selections) to the underlying macrocell are performed, or a loss of service occurs if macrocell coverage is unavailable.

The aim of the study is to automatically generate algorithms for controlling the pilot power of femtocells in any network. Addressing this type of real-world dynamic problem is an opportunity for genetic programming (GP) O'Neill et al. [27]. We use the grammatical GP [24] method called grammatical evolution (GE) [9, 29]. GE differs from GP by using a grammar, by its ability to solve typed domain problems, the use of a genotype-to-phenotype map, different genetic operators and a different chromosomal bound (length instead of depth). Previously, Hemberg et al. [13] adopted GE and generated mathematical functions with a symbolic regression grammar. In an earlier study Ho et al. [14] used GP with a set of conditional statements. Both studies focused on the feasibility of generating solutions and used the same femtocell network scenario for training and testing the solutions. We expand on these studies by comparing different grammars for creating solutions and ask the following questions:

- How does a conditional statement grammar (CG) compare to a symbolic regression statement grammar (SRG) and a combination of conditional

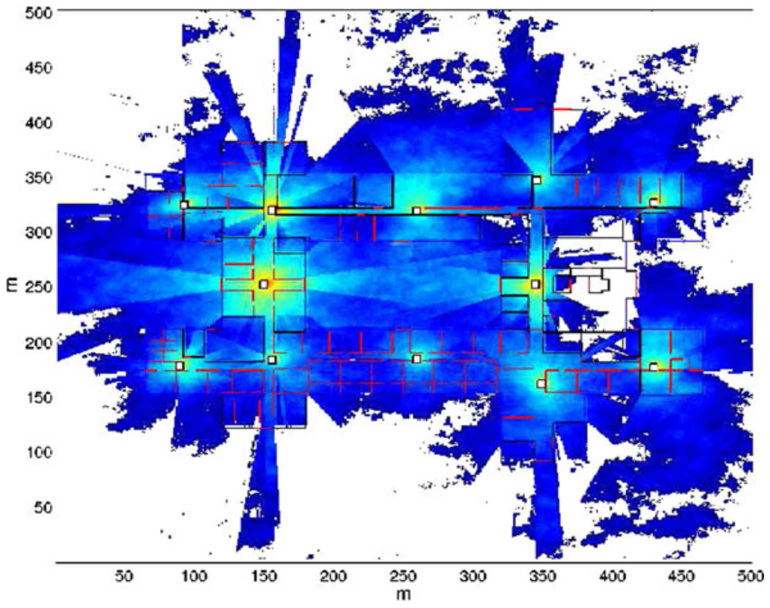

Fig. 1 Coverage of a femtocell setup for an office environment with 12 BSs, the *colour* shows the pilot channel power in dBm. The BSs are in their final state of a simulation of an evolved solution (Color figure online)

statements and symbolic regression grammar (SRCG) in terms of performance and other benefits?

- How do the evolved algorithms generalise over multiple scenarios?

Our contribution lies in the investigation of the femtocell coverage problem, the comparison of the novel and different grammars. We further introduce additional scenarios, perform an analysis of the system and employ a novel fitness function to drive the evolution. Moreover, it is shown that the best evolved solutions perform better on the test set than the simple man made approach of optimizing fixed pilot power by enumeration.

The following sections will first give a background to the GE grammars and femtocells, as well as to GE in Sect. 2 Then, in Sect. 3 the experiments are presented. The results are shown in Sect. 4, and in Sect. 5 there is a discussion regarding the setup and the results. Finally, in Sect. 6 there are conclusions and future work.

2 Background

Here the femtocell coverage problem, previous work and a preliminary analysis of the femtocell problem representation and grammatical evolution are described. All these areas aid the setup and comparison of grammars used for generating pilot power control algorithms.

2.1 Femtocell coverage problem description

We consider an intended area of coverage, e.g. an office environment as in Fig. 1, where a group of femtocells is deployed to jointly provide end-user services. The problem addresses distributed coverage optimisation by adjusting the pilot power of the BSs in order to alter the coverage of the femtocells and satisfy the following objectives:

Mobility To minimise mobility events (handovers) between femtocells and macrocells within the femtocell group's intended area of coverage.

Load To balance the load amongst the femtocells in the group to prevent overloading or under-utilisation.

Leakage To minimise the leakage of the femtocell group's coverage outside its intended area of coverage.

We believe that there are common characteristics in each scenario that are possible to capture. However, in order to achieve accuracy, complex predictors might be necessary [2]. Thus, the aim is to create algorithms that are able to generalize the expression needed to control the pilot powers of the BSs. When evaluating solutions in a simulation scenario, this can be seen as multi-dimensional data fitting.

Moreover, constructing models from observed data is a fundamental problem in science [22]. The femtocell problem can be classified as a computationally expensive multi-objective problem [36] with conflicting objectives. For example, increasing the pilot power in order to improve the coverage of a femtocell would reduce the amount of mobility events, but doing so may also increase the load of the femtocell and increase the leakage. Consequently, it is necessary to balance the requirements of the objectives. Further, the importance of the objectives depends on the operator's priorities. Finally, both the optimal pilot power and the interactions between the femtocells are unknown. This creates two additional problems, i.e. to determine a realistic simulation model and suitable fitness functions.

Now, to explain further in a more compact notation. For each scenario at each time step, $t \leq T, t \in \mathbb{N}$ there is at least one optimal pilot power configuration, $\rho^*_{ti}, \rho_{min} \leq \rho^*_{ti} \leq \rho_{max}, \rho \in \mathbb{R}$ for each BS, $i \leq n, i \in \mathbb{N}$. A scenario generates the pilot power matrix containing the pilot power of each station at each time step

$$\mathbf{P} = \begin{pmatrix} \rho_{01} & \cdots & \rho_{0n} \\ \vdots & \ddots & \\ \rho_{T1} & \cdots & \rho_{Tn} \end{pmatrix}$$

How ρ_{ti} is created is determined by the femtocells' pilot power control algorithm, $a \in A$, A is the set of possible control algorithms. We assume that it is possible to create an expression, e.g. function or algorithm, which on multiple scenarios has a pilot power close enough, δ_ρ to an acceptable pilot power, $\hat{\mathbf{P}}$, in all scenarios, $\{\forall s \in S : |\mathbf{P} - \hat{\mathbf{P}}| < \delta_\rho\}$, S is the set of scenarios.

The algorithm's decision on pilot power is mapped from pilot power to output to fitness. The output for mobility (M), load (L) and leakage (Le) are functions of the femtocells' pilot power, ρ and the scenario, s. The functions can be written as: Mobility output $o_M : \mathbb{R} \times S \rightarrow \mathbb{R}, o_M(\rho, s)$, Load output $o_L : \mathbb{R} \times S \rightarrow \mathbb{R}, o_L(\rho, s)$ and Leakage output $o_{Le} : \mathbb{R} \times S \rightarrow \mathbb{R}, o_{Le}(\rho, s)$. The outputs are the components of the fitness, $\mathbf{f} = [f_M, f_L, f_{Le}], f \in \mathbb{R}$. The fitness functions can be written as: Mobility fitness $f_M : \mathbb{R} \times S \rightarrow \mathbb{R}, f_M(o_M, s)$, Load fitness $f_L : \mathbb{R} \times S \rightarrow \mathbb{R}, f_L(o_L, s)$ and Leakage fitness $f_{Le} : \mathbb{R} \times S \rightarrow \mathbb{R}, f_{Le}(o_{Le}, s)$. We are optimizing the fitness for the femtocell control algorithms on the scenarios, hence

$$\mathbf{f}^* = \arg\max_{x \in A} \mathbf{f}(x, S)$$

In the next section we describe previous work on evolutionary computation (EC) and GE in the telecommunication domain.

2.2 Previous work

There have been previous studies of applying EC to telecommunication problems [1]. But only a few specifically regarded femtocell coverage algorithms and EC. One used GP [14] and another used GE [13] before we extend these studies. Most related work regarding cellular coverage optimisation in the literature deals with centralised computation methods [10, 34]. For example, they describe the calculation of parameters such as the number and locations of BSs, pilot channel transmit powers or antenna configurations using a central server running an optimisation algorithm. Many studies also focus on determining the optimal BS numbers or placements to achieve the operator's quality of service or coverage target. This approach is not always practical because network design is restricted by BS placements. Instead, a more realistic approach is to optimise the configuration of cellular networks where the locations of the BSs have been fixed.

One example of self-configuration and self-optimisation capability in femtocell deployments is coverage optimisation. The aim of coverage optimisation in residential femtocell deployments is to ensure that leakage of coverage by a single femtocell into public spaces is minimised while at the same time maximising indoor coverage [5, 15]. The methods for residential deployment are not applicable to enterprise environments [14]. Furthermore, there are studies which do not use EC, e.g. Jo et al. [19] investigate self-optimized coverage coordination in femtocell networks. The transmit power is adjusted based on the downlink signal and interference power statistics. A Monte Carlo simulation with static users uniformly distributed verifies the algorithm. A similar study is performed by Mhiri et al. [25] who create a power management algorithm by hand for green femtocell networks. Similarly, Ponente and De Marinis [32] optimize the transmission power and frequency for femtocells in clustered scenarios, although they use a genetic algorithm. Our approach uses grammars, simulations with moving users and considers the load of the femtocell when adjusting the pilot power.

Previous work with GP and femtocells [14] automatically derived a distributed algorithm to dynamically optimise the coverage of a femtocell group using standard tree based GP. The resulting evolved algorithm clearly showed the ability to optimise the coverage and was able to offer increased overall network capacity compared to a fixed coverage femtocell deployment. The functions and terminal set for GP consisted of conditions with predicates checking if the load, overlap and probability of users entering a gap were over a predefined threshold, as well as combining the branches of the nodes. The terminals increased pilot power, decreased pilot power or did nothing. With GE [13] the controllers for femtocells pilot power were evolved using a symbolic regression (SR) grammar. The best evolved solutions were superior for two of the objectives compared to a fixed pilot power. Here, we further investigate and compare the GE approach to different grammars, multiple scenarios and different fitness functions. Moreover, we reduce the solution space for the conditional representation of femtocell solutions.

Work by Lewis et al. [21] with a grammar and GP enhanced IEEE802.11 distributed coordinate function. They designed active MAC layer algorithms by evolving algorithms instead of optimising values and tuning parameters, thus a wider behaviour space was searched. They used a grammar to embed domain knowledge in the algorithms they evolved. The variation of contention window sizes was explored and the results outperformed standard 802.11 behaviour on a variable sized network under standard load. Moreover, the throughput performance was comparable to the best aspects of the protocol. Hu and Goodman [17] used GP for wireless access point configuration, the results improved when they post-processed their solutions to find the minimum spanning tree. In addition, Yasuda and Sato [35] used linear GP and a pruning operator on their solutions for wireless LAN access point configuration to gain improved performance and run time speed-up.

Different grammar configurations in GE have been investigated for various other applications. O'Neill and Ryan [28] first used GE to automatically evolve caching algorithms, finding a simple caching algorithm. Murphy et al. [26] explored GE for horse gait optimisation and found that the grammars used for evolving horse gaits required domain knowledge to generate realistic animal motion. Perez et al. [30] evolved behaviour trees for Mario AI using GE. The use of a grammar simplifies the task of encoding the syntax of behaviour trees and specific tree structures can be represented. They also limited the grammar syntax in order to reduce the solution space (language). We extend the use of GE to more application areas.

In a paper by Korns [20], techniques for improving symbolic regression systems in cases where the target expression contains conditionals were examined and accuracy was increased for such conditional problems. A regression system combining standard GP with abstract expression grammars, particle swarm optimisation, differential evolution, context aware crossover and age-layered populations was tested on nine base test cases. We extend the study of conditional grammars and symbolic regression for evolving femtocell controllers, where the conditions represent input states.

This section concludes that GP methods, e.g. GE, are viable for telecommunication network optimizations. It also reveals gaps in the generation of coverage algorithms for femtocells. Next, we will further analyse the previous approaches to the femtocell problem.

2.3 Initial analysis of the femtocell problem

We study the results from work by Ho et al. [14] and previous runs in order to understand and improve the setup of our own experiments. First, we simplify the solutions and this allows us to reduce the solution space, an approach that has proved to be successful in other EC applications. For example, in order to distill free form equation from experimental data with GP, Schmidt and Lipson [33] limit the representation of the equations with an acyclic graph of 128 nodes. Forrest et al. [11] also reduce the solution space when using GP for software repair, in order to get a feasible solution space.

2.3.1 Solution simplification

One way of improving the search is to create shorter solutions, e.g. by pruning the solutions and removing redundant code. However, the solution space is still the same size and there is the additional step of simplifying solutions. We studied some GP solutions from some initial runs where we used the functions and terminals presented in Table 1 and the fitness functions from Sect. 3.3 averaged over the low, medium and high load scenarios from Sect. 3.1, with the same GP setup as in Ho et al. [14]. From these results it is possible to see that the average fitness improves rapidly in the first four generations and then the fitness converges. Besides, the solutions are very bloated, they increase in size without improving fitness.

Simplification of a GP solution is made by consolidating statements. One step is removing redundant and conflicting commands, e.g. `combine3 (increasepow, decreasepow, donothing)` can be removed. It is also possible to delete branches that will never execute, e.g. `if_ho_higher(if_ho_higher (increasepow, decreasepow), decreasepow)` reduces to `if_ho_higher (increasepow, decreasepow)` since the the nested conditional will always return the same value. Further, the order of the statements is unimportant for the femtocell problem. Finally, the simplified solution is also easier to read and interpret for humans. In Fig. 2 the original best GP solution, called GP1, is shown and the simplified GP1 solution, with a reduction from 74 nodes to 11, is shown in Fig. 3.

The GP1 solution works as follows: the pilot power is changed by $+1$ dBm if there are gaps but no leakage and no overload, the pilot power is changed by -1 dBm if there is leakage, and if there is leakage and a gap the pilot power is changed by -2 dBm. Thus, the pilot power will always be decreased when there is leakage.

Table 1 GP functions and terminals

Type	Name	Description
Function	`if_ho_higher`	Check if $M > MT$, $MT = 0$
Function	`if_load_higher`	Check if $L > LT$, $LT = 7$
Function	`if_macro_requests_higher`	Check if $Le > LeT$, $LeT = 0$
Function	`combine2`	Execute branches 1 and 2 consecutively
Function	`combine3`	Execute branches 1, 2 and 3 consecutively
Terminal	`increasepow`	$\rho = \rho + C$, $C = 1$ dBm
Terminal	`decreasepow`	$\rho = \rho - C$, $C = 1$ dBm
Terminal	`donothing`	Do nothing

M mobility, *L* load, *Le* leakage and ρ pilot power

```
if_ho_higher(if_macromob_higher(combine2(decreasepow,decreasepow),
if_load_higher(if_load_higher(donothing,increasepow),if_ho_higher(
increasepow,if_load_higher(if_macromob_higher(if_ho_higher(decreasepow,
increasepow),if_macromob_higher(donothing,donothing)),combine3(combine2(
donothing,decreasepow),combine3(decreasepow,decreasepow,decreasepow),
combine3(increasepow,donothing,donothing)))))),combine2(if_ho_higher(
combine2(decreasepow,combine2(if_ho_higher(decreasepow,decreasepow),
increasepow)),combine2(if_ho_higher(if_load_higher(if_load_higher(
decreasepow,increasepow),if_macromob_higher(increasepow,increasepow)),
if_ho_higher(combine3(increasepow,donothing,donothing),increasepow)),
combine3(if_macromob_higher(combine2(decreasepow,donothing),if_ho_higher(
donothing,donothing)),combine2(if_ho_higher(decreasepow,decreasepow),
combine3(decreasepow,donothing,increasepow)),donothing))),donothing))
```

Fig. 2 Original GP1 solution

2.3.2 Representation simplification

There are redundancies in the solution representation of Table 1, here different solutions have the same behaviour, i.e. there is a many-to-one mapping from the solution space to the pilot power space. One way of improving performance would be to reduce the solution space by using a representation that produces fewer redundant solutions. For example, in the case of conditional statements this could be done by removing the possibility of freely choosing conditional statements and instead enforcing a truth table which assigns a pilot power change action to each case. We divide the search space into different states by using the conditions. There are 2^n rows for n conditions with the set of actions, a, the number of possible solutions in the search space is $|a|^{2^n}$. Using Table 2 we can devise methods for how to set the action for each row. The most general method is to use the SRCG and for each row in the table to generate real numbered pilot power values (see Fig. 7). This allows exploration of non-linear relations between the input variables, and the state division will hopefully guide the search, but will not reduce the solution or pilot power space. Another approach is to discretize the pilot power space and use constants as in CG.

2.3.3 Search space analysis

We also study the possible search space in the simulation in order to improve our representation to cover only possible pilot power settings. The BS pilot power is cut off at $\rho_{max} = 11$, $\rho_{min} = -50$. For example, if there are four action possibilities and three inputs, $-2 \leq a \leq 1, a \in \mathbb{Z}$ and $n = 3$ then the solutions can be represented by a bit-string of length 16 and the solution space is $4^{2^3} = 65{,}536$ solutions. For integer pilot power settings $|a| = |\rho_{max} - \rho_{min}| = 61$, this gives the search space size for the conditional statement as $61^8 = 1.917e + 17$. A tree where all nodes have two children, i.e. a binary tree, of depth eight is required for a solution which can increase or decrease maximally for each condition.

For the representation in Table 1 the number of trees are all the shapes with all the node combinations. The number of shapes of a binary tree with $n + 1$ leaves is given by the Catalan number $C_n = \frac{(2n)!}{(n+1)!n!}$ [31]. For each tree shape there are ϕ

Fig. 3 Simplified GP1 solution

```
if_ho_higher(
  if_macromob_higher(
    combine2(decreasepow, decreasepow
    ),
    if_load_higher(
      donothing,
      increasepow
    )
  ),
  if_macromob_higher(
    decreasepow,
    donothing,
  )
)
```

Table 2 The actions of conditional statements for the evolved control algorithm

Action	Mobility	Load	Leak	GP 1
a_1	True	True	True	-2
a_2	True	True	False	0
a_3	True	False	True	-2
a_4	True	False	False	$+1$
a_5	False	True	True	-1
a_6	False	True	False	0
a_7	False	False	True	-1
a_8	False	False	False	0

$(n) = |F|^n$ internal node combinations, F is the function set, and $\tau(l) = |T|^l$ are leaf combinations, T is the terminal set. In Ho et al. [14] the depth limit was eight and F and T are in Table 1. The max arity is two if we ignore `combine3`, thus the total number of tree shapes is $\sum_{i=o}^{l=255} C_n$ and the total number of combinations of shapes and contents is $\sum_{i=1}^{l=255} C_i^{\phi(i-1)^{\tau(i)}}$ which is larger than the solution space of 61^8 for the conditional setup. This shows that there are, with a conservative calculation, many more evolved solutions compared to possible solutions in the pilot power space. Hence, a grammar that has fewer redundant solutions is designed (see Fig. 5). Next, the grammar based GP approach called GE is explained.

2.4 Grammatical evolution

GE [9, 29] is a grammar-based form of GP [24]. It is inspired by representations in molecular biology and combines this with formal grammars. The GE system is flexible and allows the use of alternative search strategies, whether evolutionary, deterministic or other. This system also includes the ability to bias the search by changing the grammar. Since a grammar is used to describe the structures that are generated by GE, editing the grammar modifies the output structures. The genotype-phenotype (input-output) mapping means that GE allows search operations to be performed on any representation in the algorithm.

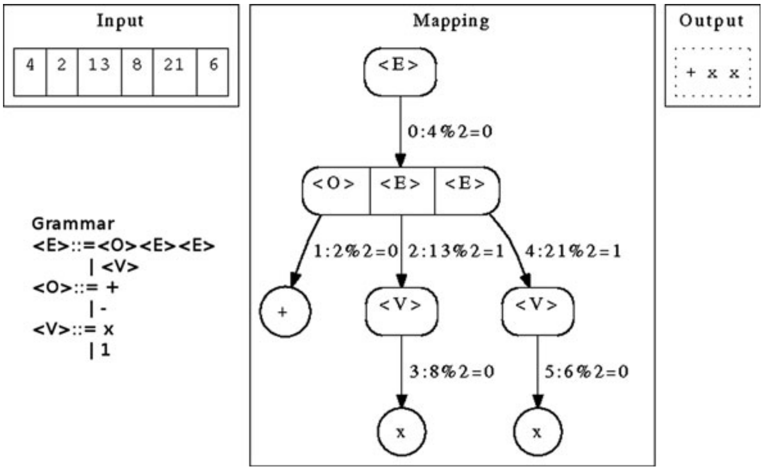

Fig. 4 Example of GE genotype-to-phenotype mapping. The derivation order, codon value and production choice are shown to the right of the arrows, e.g. from the start symbol 0:4%2 = 0. Input is the genotype and output is the phenotype

```
<CODE> ::= if gt(my_handover, MT)
             if gt(my_load, LT)
               if gt(my_macro_requests, LeT)
                 <function>
               else
                 <function>
             else if gt(my_macro_requests, LeT)
                 <function>
               else
                 <function>
           else if gt(my_load, LT)
               if gt(my_macro_requests, LeT)
                 <function>
               else
                 <function>
             else if gt(my_macro_requests, LeT)
                 <function>
               else
                 <function>
<function> ::= <terminal><function> | <terminal>
<terminal> ::= my_power = increase_power(my_power);
             | my_power = decrease_power(my_power);
             | my_power = do_nothing(my_power);
```

Fig. 5 Conditional statement grammar (CG) creates variable length solutions, <function> is recursive

In GE, the grammar mapping uses a context free grammar, in BNF-format, which is a four tuple $G = (N, T, R, S)$:

1. N is a finite non-empty set of non-terminal symbols
2. T is a finite non-empty set of terminal symbols and $N \cap T = \emptyset$, the empty set

3. R is a finite set of production rules of the form $R : N \mapsto V : A \mapsto \alpha$ or (A, α) where $A \in N$ and $\alpha \in V. V$ is the set of all strings constructed from $N \cup T$ and $R \subseteq N \times V, R \neq \emptyset$
4. S is the start symbol, $S \in N$.

The genotype is used to map the start symbol into a sentence, by the BNF-grammar. Input (codon) is read from the genotype and a corresponding output symbol is generated. A function selects a production choice by taking the current codon's integer value modulo the number of production choices of the current rule. The genotype is read from left to right, and the codon to be read is shifted to the right every time the current rule has more than one production. The derivation sequence is also expanded from left to right (depth-first). In Fig. 4 an example of a genotype generating a function is shown.

The steps in a single generation, steps 2–6 are repeated, of the GE algorithm used in the experiments are:

1. *Initialisation* The genotypes of the initial solutions are generated with the ramped half-half method.
2. *Mapping* A BNF-grammar is used: (a) Integer to String translation where the grammar maps integer values to a sentential form. (b) When the end of the genotype is reached and the output contains non-terminal symbols it wraps and is read from the start again. Finally, if there still are non-terminals in the output the individual is reinitialised.
3. *Evaluation* The individual solutions are evaluated in the simulation. Invalid solutions are reinitialised.
4. *Selection* Some individuals from the current population are included in a selected population using a tournament selection.
5. *Variation operators* Individuals are modified by one point crossover and nodal mutation.
6. *Replacement* The new population is created from best ranked solutions of the selected population and current population, based on the pareto dominance of the fitness functions.

The femtocell problem, previous work, initial studies and GE have been described in this section. The next section presents the experiments used for comparing the performance of the grammars on a number of femtocell scenarios.

3 Experiments

We aim to automatically generate pilot power control algorithms for femtocell coverage. For larger cells, such as macrocells, each cell is manually configured to separate pilot power values in a time consuming manual procedure. The macrocell configuration approach is unfeasible for femtocells, due to the associated high operational expenses. The current state-of-the-art for femtocells in enterprise deployment is to use a fixed pilot power for all femtocells. For example, the use of max pilot power guarantees the coverage objective to be met according to the capabilities of the network, without more information it is not possible to guarantee the other objectives.

For the automatically evolved algorithms investigated here, three grammars are tested on multiple scenarios, the performance is compared and the behaviour is analysed. The grammars used generated conditional solutions, symbolic regression solutions and conditional solutions containing symbolic regression. The scenario with 12 BSs in an office building, O12 (see Fig. 1) was used for training. Then the best performing solutions were picked out and evaluated on test scenarios. The variation of scenarios increases the robustness of the solutions, e.g. solutions which do not experience any overloading have an unknown behaviour when overloaded, since there is no evolutionary pressure to penalise "bad" behaviour in this state. This methodology was adapted, since it is very time consuming to run scenarios. One evaluation on one Intel i7 2.93 GHz processor core takes approximately 10 min.

In regard to comparing the performance of the grammars, Daida et al. [6] discuss the challenges of making comparisons in GP and Hoai et al. [16] describe how to compare GP systems. Our goal is to identify a good method for solving the femtocell network problem, not to dismiss the methods themselves. In the experiments, not each grammar has used the same number of fitness evaluations to reach the solution, which complicates a comparison of average fitness over fitness evaluations between the grammars. Furthermore, the femtocell network problem is not the ideal problem for benchmarking algorithmic performance, since the optimum is unknown. In order to facilitate comparisons we use engineering values to measure the performance as well. Therefore, comparisons of results from different fitness functions are possible. As a baseline we compare the evolved solutions with two manual approaches, a fixed pilot power setting at max pilot power for all the BSs, as used by Ho et al. [14], and fixed pilot power for each BS optimized by enumeration.

The simulation model, fitness function and grammars are described in the following sections.

3.1 Femtocell simulation model

A realistic simulation is needed in order to evaluate a pilot power control algorithm which would be applicable in hardware. The simulation consists of the physical environment with user movement, the load model and the radio propagation model. All these components were varied in the test scenarios. The same population movement model was used, but hot-spots and way points were configured differently for some scenarios.

In the user mobility and traffic model the users move to predefined way points on the map at a speed of 1 ms^{-1}, spending some time in a way point before moving to another way point. At the start users are randomly placed, in total 50 (low), 200 (medium) and 400 (high) users are modelled. Each user has a voice traffic model which produces 0.2 Erlangs of traffic. When evaluating an algorithm, the scenario simulates 24 h of operation time, with the algorithm adjusting the femtocell pilot power after collecting statistics for 30 min. The algorithm start time for each femtocell is randomly dithered to avoid synchronous updates. Each femtocell's initial pilot channel power is set to $\rho_0 = -30$ dBm, $-50 \leq \rho \leq 11$. In order to keep the users connected to the femtocell network for as long as possible, femtocell to macrocell handovers are triggered when a user terminal's pilot channel receive power from the best femtocell goes below -100 dBm. Outside cell users move east–west and west–east on the north and south edges in the scenario. When the signal leak is strong enough the outside user will request a handover to the femtocell and a rejection is recorded. The outside user will try to connect once to each leaking femtocell, regardless of the leakage strength.

Office (*O12, O8, O4*) The simulation scenario office environment is shown in Fig. 1. There are versions with 12, 8 and 4 femtocells for the different configurations, e.g. the training scenario in the office environment with 12 BSs and medium load is denoted O12m. The coordinates in O4 have been slightly altered compared to O8 and O12 by moving the BSs closer to the walls.

The building is an office with cubicles, closed meeting rooms, and toilets. The exterior of the building is mainly glass and the interior is mostly light interior walls and cubicle partitions. There are four stairwells at each corner with thick concrete walls. The locations of the femtocells are spaced fairly evenly apart, and done without any cell surveying. This reflects a plug-and-play deployment where some heuristic has been used in the deployment, i.e. the femtocells are not placed too closely to each other. This plug-and-play femtocell deployment is realistic, but can be sub-optimal due to the lack of exhaustive cell planning. In the simulation each femtocell has a maximum capacity of eight voice calls, and a macrocell underlay coverage is assumed. A path loss map is generated for the 450 m $\times$ 500 m area for each femtocell. For shorter distances the path loss (dB) at d (m) from a BS is modelled as $38.5 + 20\log_{10}(d) + PL_{walls}$, with a smooth transition to $28 + 35\log_{10}(d) + PL_{walls}$ in all other cases. In addition, a correlated shadow fading with a standard deviation of 8 dB and spatial correlation of $r(x) = e^{x/20}$ for a distance of x in meters is considered. The assumed transmission losses for the explicit building model are a function of the incident angle. The model is partly developed by the authors and is proprietary to Bell labs.

Outdoor (*Od4*) There are no walls. The four BSs have the same location as in O4.

Cross (*C5*) There are five femtocells and the walls, way points and hot-spots are different. The way points are set to explicitly model the need for load balancing. Moreover, there is a different path loss model where signals bounce off the walls.

3.2 Grammatical evolution setup

The aim is to generate an algorithm that increases or decreases the pilot power of the BS given some inputs. The inputs are load, mobility, leakage and pilot power. Three different grammars are used, the CG uses conditional statements, while the SRG uses mathematical functions and constants and the SRCG uses both.

3.2.1 GE grammars

The GE setup emphasises the benefit of using a grammatical representation. It is straightforward to combine the CG and SRG to create a hybrid SRCG. The aim with

```
<CODE> ::= value = <expr_0>;
<expr_0> ::= (<expr><op><expr>) | <pre-op>
<expr> ::= (<expr><op><expr>) | <var> | <pre-op> | <pre-op_step> | <pre-op_monotone>
<op> ::= + | - | .* | ./ | .^
<pre-op> ::= sin(real(<expr>)) | cos(real(<expr>)) | log(real(<expr>)) | tan(real(<expr>))
<pre-op_monotone> ::= exp(real(<expr>)) | uminus(<expr>)
<pre-op_step> ::= atan(<expr>) | tanh(<expr>) | sigmoid(<expr>)
<var> ::= my_power | my_load | my_handover | my_macro_requests | <cnst>
<cnst> ::= <nr><nr> | <nr> | 0.<nr><nr> | 0.<nr>
<nr> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Fig. 6 Symbolic Regression Statement Grammar (SRG) creates variable length solutions, `<expr>` is recursive in several steps

the SRCG is to use the domain knowledge of the conditional approach with the unconstrained expressions of symbolic regression.

Conditional statement grammar (*CG*) We construct a grammar using conditional statements aiming to replicate the GP behaviour in Table 1. The thresholds and the size of the increase and decrease of pilot power need to be predetermined, here the change is 1 dBm. The values were assigned in discussion with engineers using femtocell networks and a brief parameter sweep. Initial experiments with load were made to determine the threshold value and shape. The aim is to avoid overload or underutilization of the BS. The values of the thresholds are $MT = 0$, $LeT = 0$ and $LT = 7$, all thresholds are only binary, indicating gap or no gap, leakage or no leakage, and overload or no overload. A grammar that generates an algorithm which produces the states in Table 2 with unlimited pilot power changes, due to the recursive `<function>`, is shown in Fig. 5.

Symbolic regression statement grammar (*SRG*) Many functions were used to capture different behaviours. The terminals were chosen in order to give a large and explicitly unbiased search space, relying on evolution to determine the usefulness of the terminals. The `<pre-op_step>` is introduced to allow a step-like behaviour. Moreover, in order to avoid imaginary numbers, only the real valued part of the function values is used. We bias slightly towards non-monotonic solutions by using `<expr_0>` and also reduce the probability of using the recursive `<exp>`. The grammar adopted in this study is in MATLAB syntax and is presented in Fig. 6. The unary minus is `uminus` and non-recursive constant creation was used [8, 23].

Symbolic regression and conditional statement grammar (*SRCG*) To create the SRCG we combine the grammars in Figs. 5 and 6. The multiple `<var>` productions keep the grammar from "exploding" (see Harper [12]). The grammar has a bias towards expression using inputs compared to constants. It creates equations of the same form as SRG and uses predefined thresholds as in CG. Only the differences in CG and SRG are shown in Fig. 7.

Grammar differences The theoretical length of the possible solution strings is the same for all grammars and since all grammars are recursive, it is infinite. However, each setup has an explicit solution length limit. Furthermore, SRG and SRCG have an implicit limit imposed by the max call-stack depth in the MATLAB environment.

On the contrary, the space of possible pilot power values when evaluating the solution is different. The pilot power values for SRGs are real valued, $\rho \in \mathbb{R}$, and accordingly infinite. In contrast, the pilot power values for CG are finite or countably infinite since the pilot power changes are discrete, $\rho \in \mathbb{Z}$. This is reflected

```
<function> ::= my_power = <expr_0>;
<expr_0> ::= (<expr><op><expr>) | <pre-op>
<expr> ::= (<expr><op><expr>) | <var> | <var> | <var> | <var>
         | <pre-op> | <pre-op_step> | <pre-op_monotone>
```

Fig. 7 Symbolic regression and conditional statement grammar (SRCG). Only the differences between the CG (Fig. 5) and SRG (Fig. 6) are shown

in the parameters that need to be decided in the conditional grammar. Moreover, there is an explicit bias to changes of size one in the CG compared to the SRG.

The number of symbols in SRG is larger than in CG. In addition, SRG can construct solutions which are functions, dependent on multiple input thresholds. The size of the search space of possible solutions is greater. The SRCG has the most symbols and can generate the same equations as SRG, but generates multiple expansions, one for each state, hence the solutions are expected to be longer.

3.2.2 Random sampling and enumeration of static pilot power

We use random sampling and enumeration of fixed pilot power (SE) for analysing the behaviour of the simulation and creating optimized man made solutions. In order to make enumeration feasible, high fitness ranges known from when the pilot power is the same for all BSs are picked. The best fixed pilot power values are shown in Table 3. We also randomly sampled from the entire pilot power range. One problem with this approach is that the positioning of the cells needs to be maintained when deployed. Furthermore, random search and enumerating solutions cannot generalise to a different number of femtocells.

3.2.3 GE setup

A modified version of GE was used, based on GEM (http://ncra.ucd.ie/GEM/GEM.tgz). Nodal mutation is used [3], the nodal mutation operator has a superior property of locality compared to the standard GE mutation operator. In addition, nodal mutation is only applied to individuals that have not undergone crossover.

NSGA-II is used to rank the individuals according to domination, a solution is dominated if there is another solution which is better for all the fitness objectives. The top individuals from each front are used until the population is filled (see Deb et al. [7]). When reinitializing individuals the depth is picked from the distribution of depths on the first front. This is both an attempt to restrict bloat and search depths containing good solutions. In addition, all evaluated solutions are added to a tabu list. If a solution is already on the tabu list it will also be reinitialized. Furthermore, monotone solutions are not allowed. In order to reduce fitness function evaluations, CG and SRCG have a simplified measure of monotonicity compared to the SRG. The possible states of the inputs are passed into the evolved solution to see if it always increases, decreases or maintains its pilot power level. Finally, the input values in the grammars are then normalized.

To find extreme solutions and those which have uniform fitness components we use the index from Jain et al. [18], where a score of one is uniform and zero is

Table 3 Best enumerated static solutions for different scenarios

Scenario	BS pilot powers
O12	[−35, −35, −35, −25, −25, −35, −35, −35, −35, −35, −35, −25]
O8	[−40, −40, −28, −28, −36, −36, −36, −28]
O4	[0, −10, −15, −5]
Od4	[−35, −30, −35, −35]
C5	[−26, −26, −26, −26, −26]

non-uniform. $\phi(x) = \frac{(\sum_{i=0}^{n} x_i)^2}{n\sum_{i=0}^{n} x_i^2}$. We penalise the fitness function, $f(x)$ to get $f'(x)$ by multiplying it with its score, $h(x)$, where $h(x) = 1 - \phi^\circ\ f(x)$ and $f'(x) = e^{-h(x)}(1 - h(x)^{1/4})$. The implementation also allows for very skewed fitnesses, with the extreme solutions unpenalised when one of the objectives is zero, $h(x) = 1$ if $x = 0$.

The evolutionary parameter settings for the GE algorithm are presented in Table 4. Due to the long run time to evaluate each individual algorithm in the femtocell scenario, the number of fitness evaluations was limited. CG has a population size of 40 and has a max of 20 generations, since the solution space is significantly smaller than the SRGs. For SRCG the population size is 100 and has a max of 20 generations, enough to indicate the capabilities of the SRCG based search.

3.3 Fitness function

The fitness function is used by GE to determine the quality of the generated solutions when applied to the femtocell network. The functions are mobility, load and leakage. The duration of the simulation is T, the number of femtocells is N, and $\mathbf{x}$ is a vector of femtocells. Statistics of mobility, load and leakage are collected over a specified update period. These statistics are then used as inputs into the algorithm, and for calculating the fitness. The fitness function is a vector comprised of the fitness for each function, $\mathbf{f} = [f_M(\overline{M(\mathbf{h},\mathbf{r})}), f_L(\overline{L(\mathbf{x})}), f_{Le}(\overline{Le(\mathbf{y})})]$.

Engineering measures is the performance of a femtocell reported to an operator. The fitness function values only make sense to an EC practitioner. Instead, the engineering measures are used to decide if an algorithm is good enough. The measure which indicates load is the total demand served by the femtocell, in Erlangs (DSE), which should be maximized. The measure for leakage is the average number of mobility requests from a macrocell user per pass (MUR), which should be minimized. Mobility is measured by the average number of femtocell-macrocell handovers per user per hour (MPP), which should be minimized.

Mobility fitness is based on the number of handovers and relocations of users, derived from the femtocells' statistics of the mobility events involving femtocell users. During the simulation, the mobility events between femtocells and macrocells are recorded during an update period. The number of femtocell handovers is $\mathbf{h}$, macrocell handovers is $\mathbf{h}^M$, femtocell relocations are $\mathbf{r}$, and macrocell relocations is r^M. Mobility M is composed of $M_b^M(\mathbf{h},\mathbf{r}) = \sum_{t=0}^{T}\sum_{i=1}^{N} h_{it}^M + \sum_{t=0}^{T}\sum_{i=1}^{N} r_{it}^M$ and

Table 4 Parameter settings for the experiments. Values which are different for the grammars are labelled, e.g. population size for CG was 40, denoted by CG:40

Parameter	Value
Max wraps	2
Codon size	128
Population size	100, CG:40
Initialisation	Ramped half-and-half, depth 8
Generations	20, SRG:50
Tournament size	2
Crossover probability	0.5
Mutation	1 event per individual
Parsimony pressure	True
Extended nodal probability	0.5
Extended nodal tries	1000
Max used input	SRG:400, CG:100, SRCG:1000
Runs	SRG:12, CG:19, SRCG:19

$M_b(\mathbf{h},\mathbf{r}) = M_b^M(h,r) + \sum_{t=0}^{T}\sum_{i=1}^{N} h_{it} + \sum_{t=0}^{T}\sum_{i=1}^{N} r_{it}$. Mobility is the ratio of update periods where a mobility event occurs to the total number of update periods. It is maximised when there are no handovers or relocations to the macrocell underlay, and is 0 when all femtocell user handovers are to or from macrocells. The average mobility is 1 if there are no handovers or relocations, otherwise it is

$$\overline{M(\mathbf{h},\mathbf{r})} = \begin{cases} M_b^M(\mathbf{h},\mathbf{r})/M_b(\mathbf{h},\mathbf{r}) & \text{if} \quad M_b(\mathbf{h},\mathbf{r}) > 0 \\ 1 & \text{if} \quad M_b(\mathbf{h},\mathbf{r}) = 0 \end{cases}$$

The mobility fitness is calculated as $f_M = \overline{M(\mathbf{h},\mathbf{r})}$.

Load fitness is based on the ratio of the average number of times the load has been greater than a load threshold, LT, and the total load, including the macrocell. If the mean cell load during an update period exceeds this threshold, L is equal to one, else it is equal to zero. Cell load is $0 \le \mathbf{x} \le 7$ in this scenario, $LT = 7$, just below the capacity of the femtocell, as the aim is to prevent the femtocell from operating at its capacity. Total load is the load on the femtocells and the load on the macrocell, L_M.

$$L(\mathbf{x}) = \begin{cases} LT & \text{if} \quad \mathbf{x} > LT \\ \mathbf{x} & \text{if} \quad \mathbf{x} \le LT \end{cases}$$

Average load is $\overline{L(\mathbf{x})} = \sum_{t=0}^{T}\sum_{i=1}^{N} L(x_{it})/L_M(\mathbf{x}_t)$ and the fitness function $f_L = \overline{L(\mathbf{x})}$.

Leakage fitness is the number of outside users trying to use the femtocell. Leakage increases the number of unwanted users captured, which increases the signalling load to the core network. The leakage, Le is the ratio of blocked calls, $\mathbf{y}$ and the maximum number of macrocell users, C_{MU}. $0 \le \mathbf{y} \le C_{MU}$, $Le(\mathbf{y}) = \sum_i^N 1 - \mathbf{y}/\ C_{MU}$. The fitness function for leakage is $f_{Le} = Le(\mathbf{y})$.

This section has described the femtocell simulation, GE grammars, GE setup and fitness functions. The following section presents the results from the simulations using different grammars.

Table 5 Raw fitness for all the solutions averaged over the objectives and load types

Scenario	GP1	CG1	SRG1	SRG2	SRCG1	S	SE
O12	0.96	0.93	0.95	0.92	0.79	0.68	0.90
O8	0.91	0.89	0.92	0.90	0.74	0.64	0.91
O4	0.64	0.64	0.67	0.67	0.55	0.68	0.88
Od4	0.91	0.82	0.91	0.91	0.70	0.58	0.89
C5	0.93	0.89	0.92	0.92	0.69	0.60	0.93
Total	0.87	0.84	0.87	0.86	0.70	0.63	0.90

Total shows the averaged fitness on all scenarios

4 Results

We investigate different grammars used to generate controls for the pilot power in femtocell networks. The following results compare the test performance of the best solution from each grammar on the training set: CG1 (Fig. 10), SRG2 (Fig. 12), SRCG1 (Fig. 15), the static solution (S) of max power and enumerated static solutions (SE) in Table 5. In addition, the solutions for SRG on a larger training set, SRG1 (Fig. 13) and GP1 (Fig. 3) are inspected.

The results show that CG uses the fewest fitness evaluations to get relatively good solutions, as expected, since it has the smallest language and finite pilot power space. With SRG it is possible to fit the simulation in one expression without using much domain knowledge. Moreover, SRCG needs more fitness evaluations, since the search space is continuous and it needs to evolve even larger solutions than SRG. In addition, in SRCG each state has fewer samples than SRG to determine the fitness of the equation. Finally, a fixed length representation can be used with a reasonably sized search space and a reasonable discretization.

However, one issue is how to bias the domain knowledge and the search without over-fitting, and the SRG has the least explicit bias. In addition, the three grammars allow us to verify if the discretization is reasonable and to check that the state divisions are sensible. We can also see the effect of explicit bias in the form of domain knowledge.

4.1 Test scenarios

The GP1 solution compared to the SRG1 solution shows a small difference in fitness when averaging the values over all test scenarios. These two solutions were evolved using all the load types for O12 as training data. SRG1 has the lowest variance of the evolved solutions. The pilot power trace for each BS for some scenarios with medium load for SRG1 is shown in Fig. 8, the plot shows how the BSs slowly increase the pilot power and then it drops.

In Table 3 the raw fitness is shown for all the solutions. It can be seen that the fitness of the static solutions S and SE is the best on O4. Thus, static enumeration and random search perform better than the evolved solutions on this scenario, however, these solutions are not generalisable in the same way as the evolved

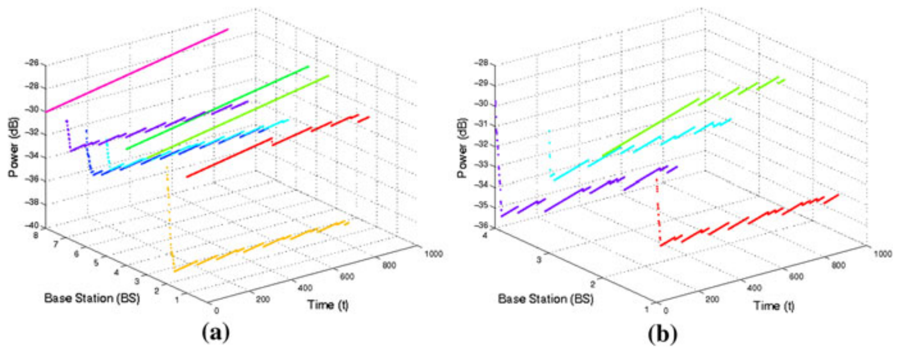

Fig. 8 SRG1 pilot power trace for each BS during the simulation on two different scenarios for load type medium. The x axis shows the time in the simulation, the y axis shows the BSs and the z axis shows the pilot power. The pilot power traces show how the BSs slowly increase the pilot power and then it drops. **a** O8, **b** Od4

solutions. The poor performance of the evolved solutions on this scenario is because the femtocells are discouraged from leaking and there are few femtocells. The low leakage can be seen by the MUR values for all solutions in Table 8, being lower on O4 than SE. Among the evolved solutions SRG1 handles the loss of BSs the best. SRCG1 and S are significantly lower ($\alpha = 0.05$) according to Wilcoxon rank sum test on the unadjusted fitness of the scenarios. For all the other solutions there are no significant differences. SRCG1 performance drops the most in the high load for all scenarios. The engineering measures for the grammars over the chosen solutions on the test and training scenarios are shown in Tables 6, 7 and 8.

The following sub sections will study the different grammars separately, showing the average best fitness over fitness evaluations. The average fitness plots filter out the extreme solutions, i.e. setting pilot power very high. It is possible for the fitness to drop since we are using the average of the average fitness objectives on the first front. The first front contains the solutions which are not dominated by any of the fitness objectives.

4.2 CG solution

The average fitness and size of the non-extreme solutions on the first front are shown in Fig. 9. We can see that the average fitness is quite high to begin with. Within quite a small number of fitness evaluations it increases even further. The variance of the fitness also decreases as the search progresses. Further evidence that solutions with few pilot power changes have good fitness can be seen in the size plot. The recursive `<function>` is the rule that changes the solutions' size, and the terminals chosen are power changes, consequently shorter solutions have fewer power changes. Thus, the CG representation needs relatively few fitness evaluations to yield quite high fitness on the training scenario.

At generation five solution CG1 was found (see Fig. 10), the solution can be reduced from 20 pilot power changing statements to 11. The CG1 solution changes the pilot power by +1 dBm only when there is a gap and no leakage and overload.

Table 6 Engineering measures for the static baseline of 11 dBm on all scenarios

Scenario	Load	S			SE		
		DSE	MUR	MPP	DSE	MUR	MPP
O12	l	7.51	8.75	0.00	7.59	1.20	0.11
O12	m	19.08	8.75	0.00	19.30	1.05	0.20
O12	h	66.89	8.74	0.00	67.23	1.05	0.22
O8	l	7.50	9.75	0.00	7.56	0.00	0.22
O8	m	18.96	9.74	0.00	19.25	0.00	0.44
O8	h	59.04	9.74	0.00	61.95	0.00	0.50
O4	l	7.39	7.00	0.00	7.48	0.91	0.00
O4	m	17.94	7.00	0.00	17.80	0.91	0.00
O4	h	31.62	6.99	0.00	31.50	1.01	0.00
Od4	l	7.27	10.25	0.00	7.28	0.50	0.02
Od4	m	17.72	10.24	0.00	17.85	0.50	0.03
Od4	h	31.55	10.24	0.00	31.64	0.50	0.04
C5	l	7.77	11.99	0.00	7.77	0.00	0.01
C5	m	18.24	11.99	0.00	18.27	0.00	0.01
C5	h	38.09	11.99	0.00	38.16	0.00	0.01

DSE, MUR and MPP are explained in Sect. 3.3

Table 7 Engineering measures for only conditional and integer solutions on all scenarios

Scenario	Load	GP1			CG1		
		DSE	MUR	MPP	DSE	MUR	MPP
O12	l	7.54	0.10	0.07	7.57	0.13	0.12
O12	m	19.23	0.17	0.08	19.38	0.24	0.16
O12	h	67.96	0.07	0.04	74.09	0.48	0.89
O8	l	7.55	0.29	0.27	7.57	0.30	0.31
O8	m	19.16	0.63	0.43	19.28	0.65	0.49
O8	h	63.64	0.11	0.35	62.41	0.26	0.93
O4	l	7.53	1.02	1.80	7.53	1.01	1.82
O4	m	18.18	0.97	3.19	18.16	0.97	3.19
O4	h	38.04	0.17	3.36	38.18	0.18	3.28
Od4	l	7.26	0.02	0.00	7.42	0.13	0.17
Od4	m	17.73	0.02	0.00	17.97	0.30	0.42
Od4	h	31.56	0.01	0.00	37.01	0.01	2.06
C5	l	7.76	0.00	0.01	7.76	0.00	0.02
C5	m	18.22	0.05	0.01	18.65	0.06	0.10
C5	h	38.57	0.00	0.02	46.27	0.00	1.39

DSE, MUR and MPP are explained in Sect. 3.3

Table 8 Engineering measures for symbolic regression solutions on all scenarios

Scenario	Load	SRG1			SRG2			SRCG1		
		DSE	MUR	MPP	DSE	MUR	MPP	DSE	MUR	MPP
O12	l	7.52	0.18	0.05	7.51	1.46	0.08	7.61	1.30	1.78
O12	m	19.14	0.16	0.06	19.11	0.25	0.13	19.36	1.40	1.45
O12	h	68.73	0.15	0.07	69.28	0.11	0.23	13.29	1.99	0.09
O8	l	7.52	0.13	0.13	7.51	0.64	0.14	7.60	1.30	2.23
O8	m	19.07	0.11	0.23	19.03	0.17	0.30	19.06	1.40	2.19
O8	h	63.74	0.12	0.27	60.09	0.07	0.41	47.54	1.30	5.85
O4	l	7.52	0.07	1.44	7.52	0.10	1.41	7.58	0.34	3.46
O4	m	18.24	0.07	2.81	18.26	0.07	2.80	16.84	0.25	5.54
O4	h	38.33	0.07	3.26	38.17	0.04	3.24	26.68	0.14	7.30
Od4	l	7.27	0.07	0.00	7.27	0.15	0.00	7.48	0.36	1.91
Od4	m	17.75	0.08	0.00	17.72	0.09	0.00	18.27	0.34	3.71
Od4	h	31.60	0.09	0.00	31.53	0.03	0.00	23.61	0.19	8.52
C5	l	7.77	0.00	0.04	7.77	0.00	0.08	7.77	0.00	1.36
C5	m	18.43	0.00	0.05	18.44	0.00	0.09	18.75	0.00	2.58
C5	h	39.59	0.00	0.06	40.69	0.00	0.09	32.81	0.00	8.83

DSE, MUR and MPP are explained in Sect. 3.3

The pilot power is changed by -1 dBm when there are leaks, but there is neither gap nor overload. When there are no gaps and overload but leakage, then the pilot power is changed by -2 dBm. It is a very readable solution, although a constant creation grammar could be used to increase the readability further, without reducing the expressiveness.

4.3 SRG solution

The average fitness and size of the non-extreme solutions on the first front are shown in Fig. 11. The early fitness values are around 0.7 and then increase to over 0.85. The number of fitness evaluations required for SRG in comparison to CG is almost 10 times more at the final evaluation, whereas the fitness averages are lower. On average there are 2,728 extra fitness evaluations for a SRG run, i.e. more than 50 % of the fitness evaluations are resulting in monotone functions. Therefore, an approximation of monotonicity, e.g. interval arithmetic, could reduce the number of fitness evaluations.

The best solution was SRG2 (see Fig. 12), found in generation 34, which contains all the inputs: mobility, load, leakage and pilot power.

4.4 Generalisation using SRG

We ran four runs of SRG testing for improved generalisation with the low, medium and high load scenario for the O12 scenario for training. We used the settings in Table 4 for SRG, except that we only ran for 30 generations. The generalization

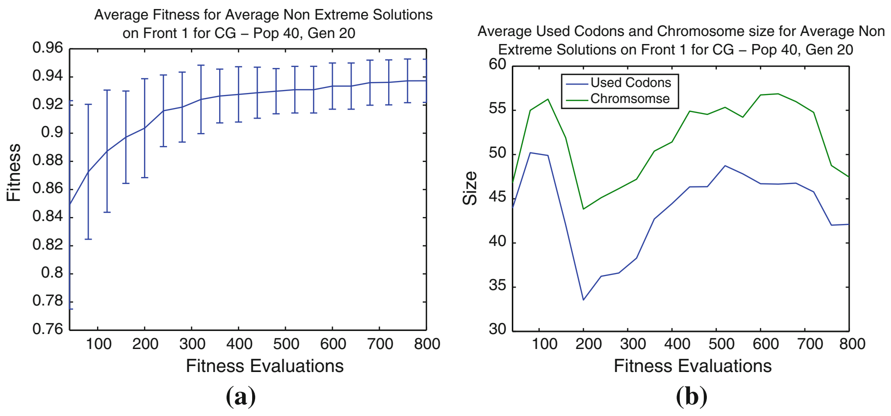

Fig. 9 Adjusted fitness performance and size for CG runs. **a** Fitness, **b** Size

```
if gt(my_handover, MT)
  if gt(my_load, LT)
    if gt(my_macro_requests, LeT)
      my_power = my_power = decrease_power(my_power);
    else
      my_power = my_power = do_nothing(my_power);
  else if gt(my_macro_requests, LeT)
      my_power = decrease_power(my_power);
    else
      my_power = increase_power(my_power);
else if gt(my_load, LT)
    if gt(my_macro_requests, LeT)
      my_power = do_nothing(my_power);
    else
      my_power = decrease_power(my_power);my_power = decrease_power(my_power);
  else if gt(my_macro_requests, LeT)
      my_power = decrease_power(my_power);
    else
      my_power = do_nothing(my_power);
```

Fig. 10 Solution CG1, simplified

requires even more extra fitness evaluations since the solution have to be non-monotone on all the load scenarios. The additional fitness evaluations are 2,631, which is almost 85 % of the number of fitness evaluations performed during a run. The average fitness over the load scenarios is adjusted.

The SRG1 solution (see Fig. 13), is only dependent on the leakage, which is transformed by several trigonometric functions, a sigmoid function and then multiplied by a constant. The power traces in Fig. 8 show that the pilot power often increases periodically until leakage reaches such a level that it is reduced again. The BS positions in each scenario show that the pilot power outputs are similar since the leakage is governing the power. In the Od4 scenario the effect of the walls is seen by the reduction in pilot power of the femtocells.

The solution ignores the load component, which is only one of three objectives. When there is not too much leakage in the simulation then the coverage is often acceptable. It is an unwanted simplification of the three objectives when there is a

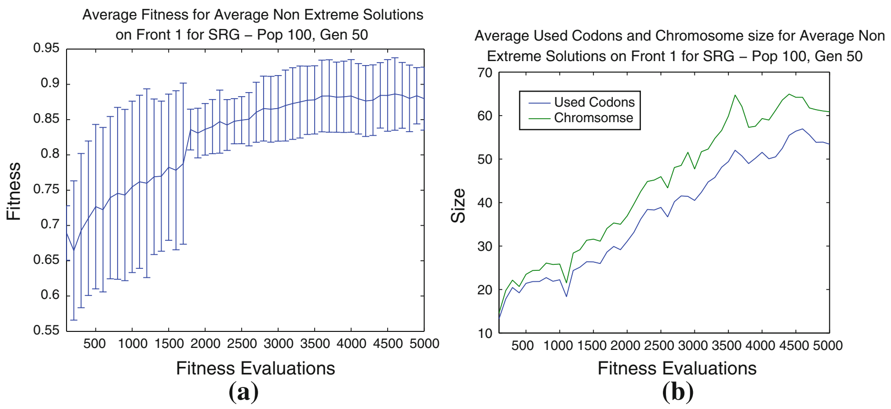

Fig. 11 Adjusted fitness performance and size for SRG runs. **a** Fitness, **b** Size

```
((cos(real(exp(real(tanh(my_load)))))-(tan(real(my_load)).*
(exp(real(my_power)).^((94+exp(real(my_handover)))./
exp(real((2-my_power))))))).*tan(real(my_macro_requests)));
```

Fig. 12 Solution SRG2

```
tan(real((cos(real(sigmoid(cos(real(atan(my_macro_requests)))))).^
tan(real(atan(uminus(exp(real(5))))))))).*log(real(tanh(3)))
```

Fig. 13 Solution SRG1

correlation between two objectives which allows the solution to reduce the impact of one. With the only input being leakage it is not possible for some BSs to cover gaps when femtocells are removed, since it would increase the leakage.

4.5 SRCG solution

The average fitness and size of the non-extreme solutions on the first front are shown in Fig. 14. The fitness has not flattened out as much as for CG and SRG, showing that the search has not converged. From the size plot it can be seen that the SRCG solutions are larger than the SRG solutions.

At generation 19 the best solution, SRCG1 was found (see Fig. 15). The solution uses more codons than the size of the chromosome and therefore it wraps around. The effect of the wrapping can be seen in the repetition of the expressions. A more refined way of handling modules and repetition might improve the results.

The results showed that the best fitness and engineering performances for the grammars are similar over both test and training scenarios. Moreover, the evolved solutions' performances is sometimes better than the simple man made approach of using fixed pilot power for each BS. SRG and SRCG require more fitness evaluations than CG, but CG requires more domain knowledge to construct. The next section further discusses the results.

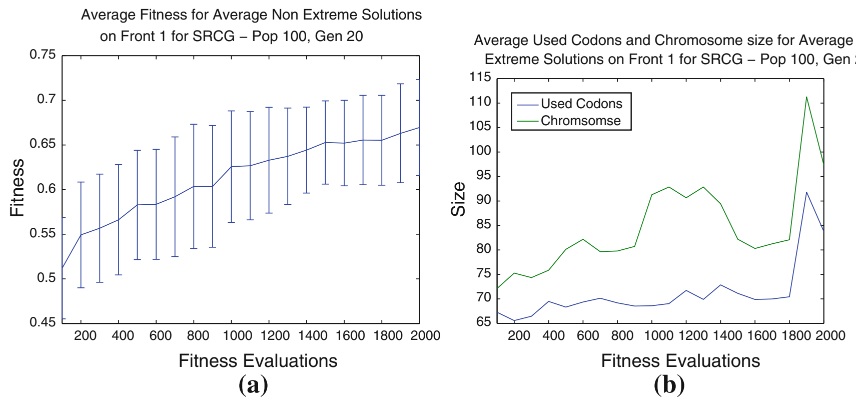

Fig. 14 Adjusted fitness performance and size for SRCG runs. **a** Fitness, **b** Size

5 Discussion

The results of the grammar comparisons raised several issues. First, the conditional functions and discrete pilot power changes are what a human engineer would anticipate as a sensible method to altering the femtocells' pilot power. The engineers find the more compact solutions easier to deal with and understand. On the other hand, the SRG setup gives the search method complete freedom to find any solution, and one of the tricks is to provide functions that allow the expression to capture the required pattern. Therefore, it might be more interesting to search for SRG and SRCG solutions.

Another issue with CG is that there are multiple good solutions in the solution space, i.e. a set of solutions with a fitness higher than a constant, $g = \{g : f(g) > C, g \in S\}$. One important issue to consider is what the ratio of good solutions is when the search space increases. Most CG solutions either do nothing or change pilot power by one, since the increments and decrements can cancel each other out. When the solution sizes are increasing, the number of pilot power changes by one grows fastest of all the changes. Therefore, if there are good solutions which require only single pilot power changes, the probability of generating them does not decrease too rapidly. The SRG and SRCG have no such bias.

From the random sampling of the scenarios it can be seen that the mobility fitness value is the most skewed, but reaches the entire fitness range with some outliers. The leakage is the most uniformly distributed value. Increasing the number of BSs lowers the median leakage as expected, since there are more stations that can leak. If the pilot power is kept roughly in the range [–40, –20] then the more BSs the higher the mobility and load, since there are more stations that can cover the area as well as avoid overload. When comparing the office environment with the Od4 enumeration the load range is larger for open spaces. This can be explained by the walls blocking the signal, which keep cells from being overloaded.

```
if gt(my_handover, MT)
  if gt(my_load, LT)
    if gt(my_macro_requests,LeT)
      my_power = sin(real(my_power));
    else
      my_power = log(real(cos(real(my_macro_requests))));
  else if gt(my_macro_requests, LeT)
      my_power = (cos(real(exp(real(sin(real(my_handover)))))).*my_macro_requests);
    else
      my_power = tan(real(sigmoid(tanh(log(real(my_macro_requests))))));
else if gt(my_load, LT)
    if gt(my_macro_requests, LeT)
      my_power = (my_power./atan((my_power.*my_power)));
    else
      my_power = log(real(cos(real(my_macro_requests))));
  else if gt(my_macro_requests, LeT)
      my_power = (cos(real(exp(real(sin(real(my_handover)))))).*my_macro_requests);
    else
      my_power = tan(real(sigmoid(tanh(log(real(my_macro_requests))))));
```

Fig. 15 Solution SRCG1

5.1 Load

There is an inherent difficulty in distributed load balancing. For example, in a scenario when a BS is overloaded and it reduces its pilot power, a gap can be created. A neighbouring BS needs to detect and then increase its pilot power to cover this gap. The problem is that the BS might still be overloaded, in addition to possible fitness penalties from lack of coverage during the time that the coverage gap exists. Therefore, evolution might find it more beneficial to ignore overloading, in order to avoid additional fitness penalties.

Furthermore, load can also be difficult to balance if there are no BSs which are capable of handling an increased load. Another reason for the difficulty of detecting overloaded states is the threshold of the load. The load fitness function creates a very sharp step for the fitness. The other states are entered if the input values are >0, while the load is >7, o_L, o_M, $o_{Le} \geq 0$. The input values are averaged over the update time period and are always >0. Mobility and leakage will always be greater than their threshold of zero if an event occurs. In contrast, the load threshold requires the load to be high most of the time.

5.2 States

We can trace the states of the algorithm in the BS, described in Sect. 2.3.2, by recording the transition between states. If we correlate the states with the conditions in the grammar we can gain information about the algorithm, otherwise, we only gather information concerning the simulation. By tracing the states we gain information regarding the input that the algorithm reacts to. The state trace helps analysing the contents of the simulation as well as explaining nonsensical statements in the algorithm. The "junk" in the algorithm is there because the state is never reached. This is unwanted and will affect the robustness of the algorithm, since when a new scenario invokes an unseen state

Figure 16 shows the state trace for some solutions on O12m, with state numbers as in Table 2. The most desirable state is eight (8) and the least desirable is one

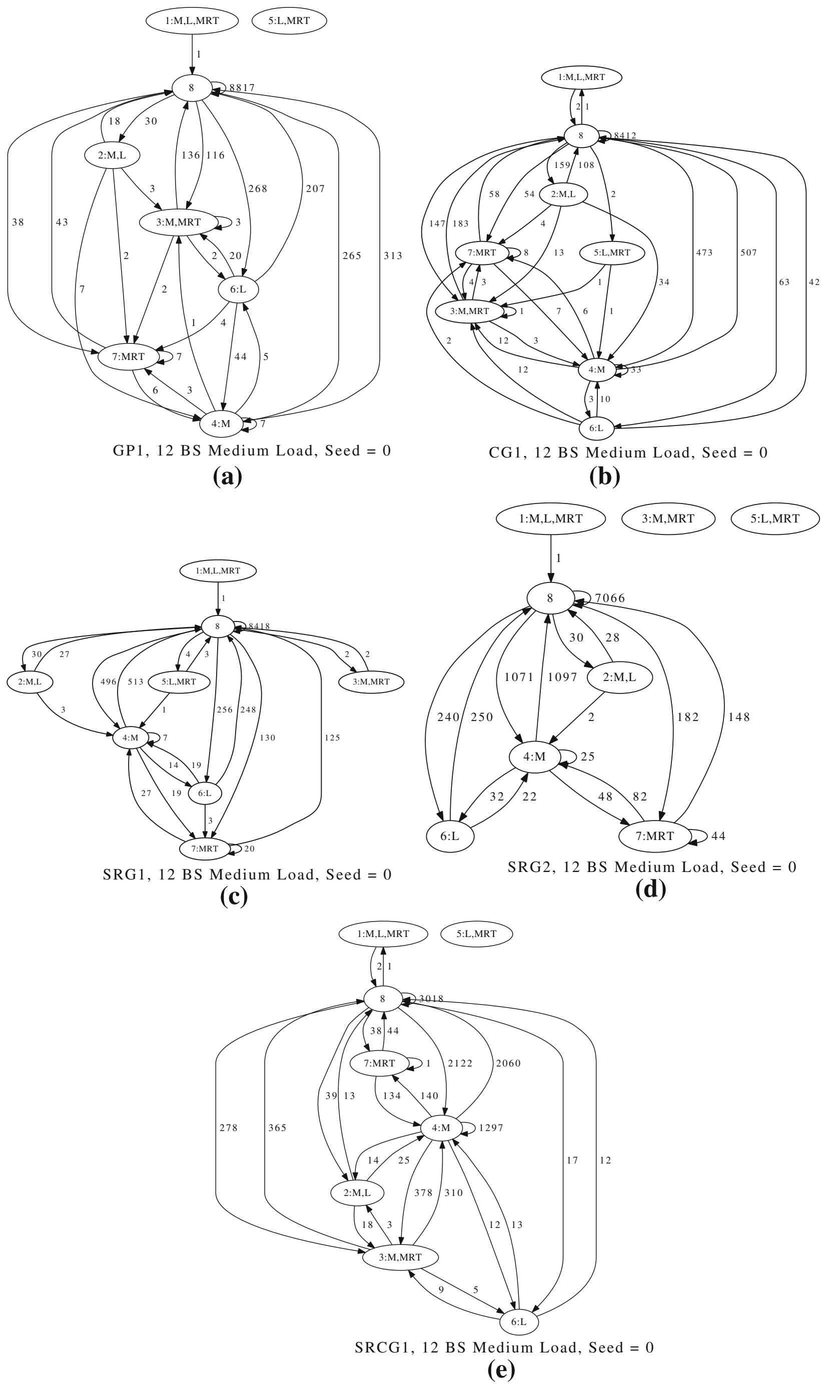
1:M,L,MRT
5:L,MRT
8
2:M,L
3:M,MRT
6:L
7:MRT
4:M
GP1, 12 BS Medium Load, Seed = 0
(a)
CG1, 12 BS Medium Load, Seed = 0
(b)
SRG1, 12 BS Medium Load, Seed = 0
(c)
SRG2, 12 BS Medium Load, Seed = 0
(d)
SRCG1, 12 BS Medium Load, Seed = 0
(e)

Fig. 16 State trace for solutions on O12m, with state numbers as in Table 2. *M*, *L*, *Le* indicate if the input is *above* a threshold. **a** GP1, **b** CG1, **c** SRG1, **d** SRG2, **e** SRCG1

(1:*M*, *L*, *Le*), which is hardly ever reach by any algorithm. The solutions all have different behaviours, e.g. GP1 has the fewest gaps, whereas GP1, SRG2 and SRCG1 do not reach all the states. However, GP1 was evolved using all load types which indicates that it is difficult to balance the load in O12. As discussed in Sect. 5.1 from an overloaded state (6:*L*) a gap (4:*M*) is reached, but a gap and overload (2:*M*, *L*) are never reached directly. The analysis shows that each algorithm behaves differently.

6 Conclusions and future work

We performed an initial study of grammars used in grammatical GP which create algorithms that control the BS pilot power in a femtocell network. The overall goal is to create a continuous online evolution of the femtocell pilot power control algorithm. We compared the performance of the different grammars and then analyzed the femtocell simulation model.

The grammars consisted of conditional statements, mathematical functions, as in symbolic regression and a combination of symbolic regression and conditional statements. The conditional grammar is a simplification of a GP representation previously used, but with a smaller solution space and significant improvement in human readability, which is demanded by the engineers. The solutions regulate the pilot power with discrete or continuous values. We also performed limited enumeration of femtocell pilot power settings and random sampling to gain further information about our femtocell network simulation model

The results showed that the best fitness and engineering performances for the grammars are similar over both test and training scenarios. In addition, the evolved solutions' performance is sometimes better than the simple man made approach of using fixed pilot power for each base station. The symbolic regression statement grammars require more configuration of the evolutionary algorithm and more fitness evaluations. But to construct the pure conditional statement grammar more domain knowledge is required.

Different grammars have different languages and a different bias to power changes. The language of CG creates solutions which perform well with the least fitness evaluations. The same language as CG, but using GP instead, generates solutions that perform equally well. The language for SRG generates solutions that perform as well as CG. SRCG has lower performance. It is the language with the most terminals and here the solution sentences are the longest in order to be valid, e.g. a SRCG solution completely identical to a SRG solution is eight times longer. Therefore, SRCG requires more fitness evaluations than SRG, which is one reason for its low performance. Finally, the content of the resulting femtocell algorithms showed that the EC methods are exploiting the simplifications in the model. The ability of EC to exploit bias in both the fitness function and the underlying model is vital for identifying the current system and can either improve the model or the EC method.

Future investigations will concern the use of more local search to create a hybrid algorithm and explore promising solutions. Another hybrid version to explore is to first find a solution which satisfies the constraints and then apply GE. There are more scenarios to test and also possibilities to refine operations on the conditional symbolic regression grammar. Moreover, it is possible to simplify solutions during the run and add more domain knowledge to SRCG. The threshold values and ranges can be explained further, e.g. use percent of load as a fitness function. In addition, the method can be applied to other constraint satisfaction problems, for example asset and sensor networks.

Acknowledgments This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

1. E. Alba, J.F. Chicano, in *Evolutionary Algorithms in Telecommunications*. IEEE Mediterranean Electrotechnical Conference (MELECON 2006), pp. 795–798. IEEE (2006)
2. L. Breiman, Statistical modeling: the two cultures. Statist. Sci. **16**(3), 199–215 (2001)
3. J. Byrne, M. O'Neill, J. McDermott, A. Brabazon, An analysis of the behaviour of mutation in grammatical evolution. Genet. Program. **6021**, 14–25 (2010)
4. V. Chandrasekhar, J. Andrews, A. Gatherer, Femtocell networks: a survey. IEEE Commun. Mag. **46**(9), 59–67 (2008)
5. H. Claussen, F. Pivit, L.T.W. Ho, Self-optimization of femtocell coverage to minimize the increase in core network mobility signalling. Bell Labs Tech. J. **14**(2), 155–183 (2009)
6. J.M. Daida, D.S. Ampy, M. Ratanasavetavadhana, H. Li, O.A. Chaudhri, in *Challenges with Verification, Repeatability, and Meaningful Comparison in Genetic Programming: Gibson's Magic*. GECCO, vol. 2, pp. 1851–1858. Citeseer (1999)
7. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. I. Dempsey, M. O'Neill, A. Brabazon, in *Grammatical Constant Creation*, ed. by D. Kalyanmoy et al. Genetic and Evolutionary Computation—GECCO-2004, part II, vol. 3103. Lecture Notes in Computer Science (Springer, Seattle, 2004), pp. 447–458
9. I. Dempsey, M. O'Neill, A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments* (Springer, Berlin, 2009)
10. D. Fagen, P.A. Vicharelli, J. Weitzen, Automated wireless coverage optimization with controlled overlap. IEEE Trans. Veh. Technol. **57**(4), 2395–2403 (2008)
11. S. Forrest, T.V. Nguyen, W. Weimer, C. Le Goues, *A Genetic Programming Approach to Automated Software Repair* (ACM, New York, NY, 2009), pp. 947–954
12. H. Robin, in *Ge, Explosive Grammars and the Lasting Legacy of Bad Initialisation*. IEEE World Congress on Computational Intelligence (WCCI 2010) (2010)
13. E. Hemberg, L. Ho, M. O'Neill, H. Claussen, in *A Symbolic Regression Approach to Manage Femtocell Coverage Using Grammatical Genetic Programming*. GECCO, pp. 639–646. ACM (2011)
14. L. Ho, I. Ashraf, H. Claussen, in *Evolving Femtocell Coverage Optimization Algorithms Using Genetic Programming*. IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications, 2009, pp. 2132–2136. IEEE (2010)
15. L.T.W. Ho, H. *Claussen, in Effects of User-Deployed, Co-channel Femtocells on the Call Drop Probability in a Residential Scenario*. IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007), pp. 1–5. IEEE (2007)
16. N.X. Hoai, RI McKay, D. Essam, H.A. Abbass, Toward an alternative comparison between different genetic programming systems. Genet. Program. **7**, 67–77 (2004)
17. J. Hu, E. Goodman, in *Wireless Access Point Configuration by Genetic Programming*. IEEE Congress on Evolutionary Computation, pp. 1178–1184 (2004)
18. R. Jain, D.M. Chiu, W.R. Hawe, in *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System* (Eastern Research Laboratory, Digital Equipment Corp., Hudson, MA, 1984)
19. H.S. Jo, C. Mun, J. Moon, J.G. Yook, Self-optimized coverage coordination in femtocell networks. IEEE Trans. Wirel. Commun. **9**(10), 2977–2982 (2010)

20. M.F. Korns, Symbolic regression of conditional target expressions. Genet. Program. Theory Pract. VII 211–228 (2010)
21. T. Lewis, N. Fanning, G. Clemo, in *Enhancing IEEE802. 11 DCF Using Genetic Programming*. IEEE 63rd Vehicular Technology Conference (VTC 2006), vol. 3, pp. 1261–1265. IEEE (2006)
22. L. Ljung, Perspectives on system identification. Annu. Rev. Control. **34**(1), 1–12 (2010)
23. M. O'Neill, I. Dempsey, A. Brabazon, C. Ryan, in *Analysis of a Digit Concatenation Approach to Constant Creation*, vol. LNCS 2610 (Springer, Essex, UK 2003), pp. 173–182
24. R.I. McKay, N.X. Hoai, P.A. Whigham, Y. Shan, M. O'Neill, Grammar-based Genetic Programming: a survey. Genet. Program. Evol. Mach. **11**(3), 365–396 (2010)
25. F. Mhiri, K. Sethom Ben Reguiga, R. Bouallegue, G. Pujolle, in *A Power Management Algorithm for Green Femtocell Networks*. The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2011, pp. 45–49. IEEE (2011)
26. J. Murphy, M. O'Neill, H. Carr, Exploring grammatical evolution for horse gait optimisation. Genet. Program. **5481**, 183–194 (2009)
27. M. O'Neill, L. Vanneschi, S. Gustafson, W. Banzhaf, Open issues in genetic programming. Genet. Program. Evolv. Mach. **11**(3), 339–363 (2010)
28. M. O'Neill, C. Ryan, Automatic Generation of Caching Algorithms, ed. by K. Miettinen et al. Evolutionary Algorithms in Engineering and Computer Science (Wiley, Finland, 1999), pp. 127–134
29. M. O'Neill, C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language* (Kluwer, Norwell, 2003)
30. D. Perez, M. Nicolau, M. O'Neill, A. Brabazon, Evolving behaviour trees for the mario ai competition using grammatical evolution. Appl. Evol. Comput. **6624**, 123–132 (2011)
31. R. Poli, W.B. Langdon, S. Dignum, in *On the Limiting Distribution of Program Sizes in Tree-Based Genetic Programming*. Technical Report CSM-464, Department of Computer Science, University of Essex (2006)
32. G.I. Ponente, E. De Marinis, in *Femtocell System Optimization by Genetic Algorithm in Clustered Scenarios*. Future Network and Mobile Summit (FutureNetw), 2011, pp. 1–9. IEEE (2011)
33. M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data. Science **324**(5923), 81 (2009)
34. I. Siomina, P. Varbrand, Automated optimization of service coverage and base station antenna configuration in umts networks. IEEE Commun. Wirel. **13**(6), 16–25 (2006)
35. Y. Yasuda, Y. Sato, in *Using Genetic Programming to Improve the Performance of Wireless LAN Access Point Configuration*, ed. by The Long Pham et al. Proceedings of the Third Asian-Pacific Workshop on Genetic Programming, pp. 57–68. Vietnam (2006)
36. A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011)