Evolving Femtocell Algorithms with Dynamic and Stationary Training Scenarios

Erik Hemberg[1], Lester Ho[2], Michael O'Neill[1], and Holger Claussen[2]

[1] Natural Computing Research & Applications Group
Complex & Adaptive Systems Laboratory
School of Computer Science & Informatics
University College Dublin
erik.hemberg@ucd.ie, m.oneill@ucd.ie
[2] Bell Laboratories
Alcatel-Lucent
Dublin
{lester.ho,holger.claussen}@alcatel-lucent.com

Abstract. We analyse the impact of dynamic training scenarios when evolving algorithms for femtocells, which are low power, low-cost, user-deployed cellular base stations. Performance is benchmarked against an alternative stationary training strategy where all scenarios are presented to each individual in the evolving population during each fitness evaluation. In the dynamic setup, different training scenarios are gradually exposed to the population over successive generations. The results show that the solutions evolved using the stationary training scenarios have the best out-of-sample performance. Moreover, the use of a grammar which produces discrete changes to the pilot power generate better solutions on the training and out-of-sample scenarios.

1 Introduction

Femtocells are low power, low-cost, user-deployed cellular base stations (BS), which operate in dynamic environments. A significant issue facing the developers of the algorithms which control the behaviour of femtocells is how best to design the algorithms to handle these unforeseen, dynamic environments. In previous studies [12, 11] we have successfully examined the suitability of Genetic Programming (GP), and a grammar-based form of GP [14], Grammatical Evolution (GE) [6], to generate control algorithms for these devices. In these earlier studies a predefined, static set of scenarios are exposed to the evolving population to determine the quality of the evolving solutions.

Our aim in this study is to examine the impact of the training scenarios employed on the quality of the evolved solutions. More specifically we ask:

– *Is there a difference in the robustness of solutions (out-of-sample) based on the use of stationary versus dynamic training scenarios?*

The remainder of the paper is structured as follows. In Sect. 2 the femtocell problem is described. Experiments and results are in Sect. 3 and 4. Finally, the conclusion and future work is presented in Sect. 5.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 518–527, 2012.

2 The Femtocell Problem

As is the case with other physical network infrastructure such as base stations, there are a number of issues surrounding the optimal placement of the hardware in addition to the design of algorithms which manage the performance of hardware networked in this manner. Femtocells are low power, low-cost, **user-deployed** cellular base stations. Therefore, in the case of femtocells the designer of the software does not know *a-priori* where (and how many) femtocells might be deployed in a site.

If we consider an intended area of coverage, e.g. an office environment as shown in Fig. 1(c), where a group of femtocells is deployed to jointly provide end-user services, we focus on the problem of distributed coverage optimisation by adjusting the pilot power of the BS in order to alter the coverage of the femtocells. The objectives are:

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.

There have been previous studies of applying EC to telecommunication problems [1]. But only two specifically regarding femtocell coverage algorithms and EC, one using GP [12] and another using GE [11]. Most related work in the literature regarding cellular coverage optimisation deals with centralised computation methods [16, 8], e.g. the calculation of parameters such as the number and locations of BS, 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, and in the case of femtocells these are physically deployed by the end-user.

3 Experimental Setup

In the femtocell problem we face a number of challenges, the most pressing of which are (i) fitness evaluations are computationally expensive, and (ii) it is not clear how best to design the fitness evaluations in terms of the type and number of training scenarios presented to the evolving population. In this study we focus on understanding how to best design a fitness function by examining the robustness of solutions evolved using dynamic and stationary training scenarios. In terms of computational expense, the dynamic training scenarios are potentially attractive as less scenarios are presented to each individual of the population, thereby reducing the evolutionary algorithms run time. In addition, there are potentially performance gains to be achieved by adopting dynamic environments during evolutionary runs, for example, see [15]. We therefore study the robustness of solutions depending on how they have been evolved. The two approaches we use to drive evolution are:

Stationary training scenario: The fitness function employs multiple training scenarios at each generation. The training fitness is calculated as the average fitness across each training scenario presented to the individual.

Dynamic training scenario: The fitness function is comprised of a single training scenario at each generation and as the generations progress the training scenario changes.

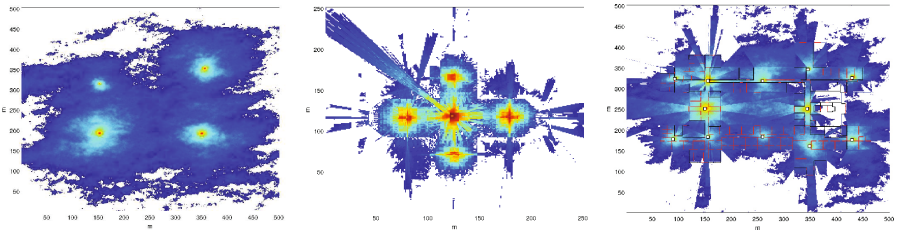
The guidance of the search towards a solution is different for each setup. The stationary setup is comprised of multiple training scenarios, where evolution is trying to find a general solution by averaging the fitness over several scenarios. The reasoning is that solutions that are too specialised will be avoided since a solution must have good fitness on all the scenarios. In contrast, the dynamic setup exposes the evolving population to a single scenario which changes over time, attempting to guide the population towards solutions which can behave well on scenarios presented over different environmental conditions. There are more assumptions and uncertainties in the dynamic scenario regarding the robustness of the solution. First, the search must be given enough evaluations in each scenario to find good solutions. Then, the next scenario needs to be similar enough to allow the search to gain advantage from existing parts in the solutions. Thus, this is a potentially powerful approach for stimulating generic parts in solutions. Although it requires the capability of the search method and setup to represent and identify general components which are preserved during the search.

3.1 Simulation Model

A user mobility and traffic model is employed, where users are initially randomly placed at way points on the map and moving at a speed of $1ms^{-1}$, spending some time at a way point before moving to another. In total 50, 200 and 400 users are modelled, in *low* (l), *medium* (m) and *high* (h) load scenarios. Each user's voice traffic model produces 0.2 Erlangs of traffic during 24 hours of simulated operation time, with the algorithm adjusting the femtocell pilot power after collecting statistics for 30 minutes. The algorithm start time for each femtocell is randomly dithered with, and the initial pilot channel power $\rho = -30$dBm, $\rho \in [-50, -49, \ldots, 11]$. Femtocell to macrocell handovers are triggered when a user terminal's pilot channel receive power from the best femtocell goes below -100dBm. Outside cell users move east-west and west-east on the north and south edges of the map. When the signal leakage is strong enough the outside user request a handover to the femtocell and a rejection is recorded. The outside user tries to connect once to each leaking femtocell when moving through the femtocell coverage.

Office (O12, O8, O4). The number of BS in the office environment are 12, 8 and 4, shown in Fig. 1. The scenario with 12 BS is denoted **O12**. In **O4** the coordinates have been slightly altered compared to the **O8** and **O12** scenarios, 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. This is a realistic plug-and-play femtocell deployment, which can be



(a) Od4 (b) C5 (c) O12

Fig. 1. Received pilot power for the **Outdoor(Od4)** 1(a), **Cross(C5)** 1(b) and **Office(O12)** 1(c)

sub-optimal due to the lack of exhaustive cell planning. In the simulation each femtocell has a maximum capacity of 8 voice calls, a macrocell underlay coverage is also assumed. A path loss map is generated for the 450m x 500m area for each femtocell. For shorter distances the PL, path loss (dB) at d (meters) from a BS is modelled as $38.5 + 20log_{10}(d) + PL_{walls}$, with a smooth transition to $28 + 35log_{10}(d) + PL_{walls}$ otherwise. A correlated shadow fading with a standard deviation of 8 dB and spatial correlation of $r(d) = e^{d/20}$. The assumed transmission losses for the explicit building model are a function of the incident angle, this model is taken from Ho et al. [12].

Outdoor (Od4). There are no walls and the BS placement is the same as in **O4**.

Cross (C5). There are walls and 5 BS. All the way points and hot-spots are different from **O12** and set to explicitly model the need for load balancing by overloading some cells and under utilizing others. Moreover, a different path loss model is used.

The training scenarios are **Od4, C5, O12** with medium load, the validation scenario is **O4l**, and the test scenarios are **O8, O4** at low, medium and high load. The dynamic setup starts with **Od4**, see Fig. 1(a). The next scenario is **C5** Fig. 1(b). The last scenario is **O12** Fig. 1(c). Thus, there is an increase in number of BS and the walls between each scenario. The stationary scenarios evaluate on all the scenarios at every iteration.

3.2 Evolutionary Algorithm

In this study we use a Matlab implementation of GE, GEM [1]. Two different grammars are tested (denoted CG and SRCG). A conditional grammar that changes the pilot power with discrete values and a conditional equation grammar changing the pilot power with continuous values calculated from generated equations. The search space is very different for the grammars. A difference in performance is to be expected with the number of fitness evaluations used. In addition, the expected result from the different scenarios would be that stationary scenarios should perform well since it is always the same underlying simulation model. The setup is the same as Hemberg et al. [11].

[1] `http://ncra.ucd.ie/GEM/GEM.tgz`

```
<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. 2. Conditional statement grammar (CG)

```
<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>
<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. 3. Symbolic Regression and Conditional Statement Grammar (SRCG). Only the differences between the CG (Fig. 2) is shown

Conditional Statement Grammar (CG). We construct a grammar using conditional statements. The thresholds and the size of the increase and decrease of power needs to be predetermined. Here the change is 1dBm and the thresholds are mobility ($MT = 0$), leakage ($LeT = 0$) and load ($LT = 7$).

Symbolic Regression and Conditional Statement Grammar (SRCG). Creates equations and uses thresholds as in CG. Only the differences in CG and SRCG are shown in Fig. 3. To create the SRCG we combine the grammar in Fig. 2. The multiple `<var>` productions keeps the grammar from "exploding", see Harper [10]. The grammar adopted in this study is in MATLAB syntax. A wide range of functions were used and only the real valued part of the function values was used. The unary minus is `uminus`.

Fitness Function. 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 duration of the simulation is T, the number of femtocells is

N, and $\mathbf{x}$ is a vector of femtocells. The fitness is a vector comprised of the fitness for each function, $\mathbf{f} = [f_M(\overline{M(h,r)}), f_L(\overline{L(\mathbf{x})}), f_{Le}(\overline{Le(\mathbf{x})})]$. The mobility objective is conflicting with load and leakage, leakage can also conflict with load. All the objectives are normalized and equally important.

Mobility fitness is the number of handovers and relocations of users. The mobility events between femtocells and macrocells are recorded for each period. The number of femtocell handovers is h, macrocell handovers is h^M, femtocell relocations is r, and macrocell relocations is r^M. Mobility, M, is the ratio of update periods where a mobility event occurs divided by the total number of update periods.

$$M_b^M(h,r) = \sum_{t=0}^{T}\sum_{i=1}^{N} h_{it}^M + \sum_{t=0}^{T}\sum_{i=1}^{N} r_{it}^M$$

$$M_b(h,r) = M_b^M(h,r) + \sum_{t=0}^{T}\sum_{i=1}^{N} h_{it} + \sum_{t=0}^{T}\sum_{i=1}^{N} r_{it}$$

The mobility fitness is maximised when there are no handovers or relocation to the macrocell underlay, and is 0 when all femtocell user handovers are to or from macrocells, otherwise

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

Load fitness has the objective that the femtocells should serve enough users. It is based on the ratio of average number of times the load has been greater than a defined maximum load threshold, LT, and the total load, including the macrocell. If the mean cell load during an update period exceeds LT then L is equal to one, else it is equal to zero. Cell load is $0 \leq \mathbf{x} \leq 8$ in this scenario, $LT = 7$, below the capacity of the femtocell, to prevent operation at full capacity. Total load is the sum of the femtocells and the macrocell, L_M.

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

Average load is $\overline{L(\mathbf{x})} = \sum_{t=0}^{T}\sum_{i=1}^{N} L(\mathbf{x}_{it})/L_M(\mathbf{x}_t)$.

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, y, to the maximum number of macrocell users, C_{MU}, $0 \leq y \leq C_{MU}$ with $Le(y) = 1 - y/C_{MU}$.

GE Parameters. The evolutionary parameter settings for the GE algorithm are presented in Table 1.

Nodal mutation [4] is used instead of the standard integer mutation. The multiple objectives are tackled with the NSGA-II, see Deb et al. [5]. When reinitializing individuals the max derivation tree depth is picked from the distribution of derivation tree depths in the first front. This is both an attempt to restrict bloat and search at derivation

Table 1. Parameter settings for the experiments (dynamic setup (DTS) stationary setup (STS))

Parameter	Value
Max wraps	2
Codon size	128
Population size	20
Initialisation	Ramped half-and-half
Initialisation depth	8
Generations	STS:10; DTS:30
Tournament size	2
Crossover probability	0.5
Mutation	1 event per individual
Parsimony pressure	True
Runs	28

tree depths where good solutions have been found. All evaluated solutions are added to a tabu list and if a solution is in the tabu list the solution will be reinitialized [11]. Furthermore, monotone solutions are not allowed, i.e. only static, increasing or decreasing power.

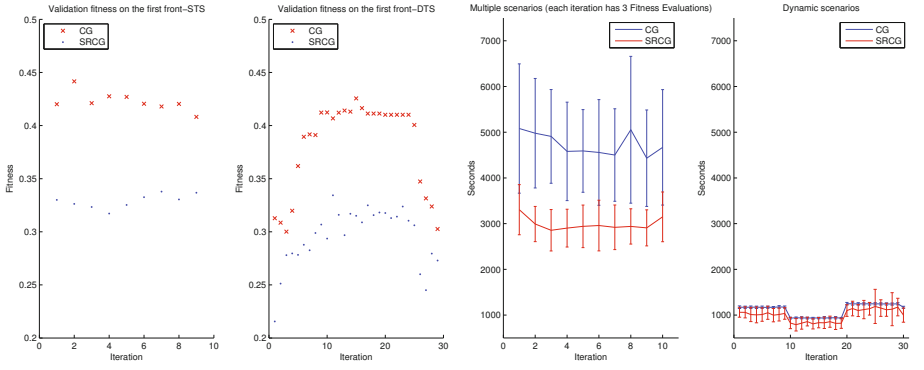
To find solutions which maximizes one objective and those which have uniform fitness components we use the method from Jain et al. [13] to modify the fitness, where a score of one is the components are uniform and zero they are non-uniform, $\phi(\mathbf{x}) = \frac{(\sum_{i=0}^{n} x_i)^2}{n \sum_{i=0}^{n} x_i^2}$. We penalise the fitness function vector, $\mathbf{f}(\mathbf{x})$ to get $\mathbf{f}'(\mathbf{x})$ by modifying it with its score, $\mathbf{h}(\mathbf{x})$, where $\mathbf{h}(\mathbf{x}) = 1 - \phi \circ \mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\mathbf{x}) = e^{-\mathbf{h}(\mathbf{x})}(1 - \mathbf{h}(\mathbf{x})^{1/4})$.

4 Results

The grammars and setups are run independently 28 times with different seeds for the pseudo-random number generator. To simplify the presentation the average of the fitness function vector is shown. Figure 4(b) outlines the results, in terms of the run time of the dynamic (DTS) versus stationary (STS) setups. We can observe substantially lower run times for the DTS. Both setups use the same number of fitness evaluations and the total run-time was also significantly different, from a t-test at a 0.05-level, for all comparisons except SRCG in DTS and STS.

With respect to the quality of solutions evolved using the different training scenarios, training results are presented in Figure 5, validation performance in Fig. 4(a), and out-of-sample performance outlined in Table 2.

The mean fitness of all objectives from the training fitness of all solutions in the population, excluding extreme solutions, progresses towards higher fitness, shown in Fig. 5. Since there are changes in the fitness function and the values are the average of the front it is possible for the fitness value to drop. The values decrease when more solutions with lower average are added to the first front. The difference between the methods is significant as can be seen by the non-overlapping error-bars. The graphs show that the representation in CG finds good solutions very fast in comparison to SRCG. The SRCG also has a larger standard deviation. Note that the graphs only show



(a) Validation fitness (b) Time

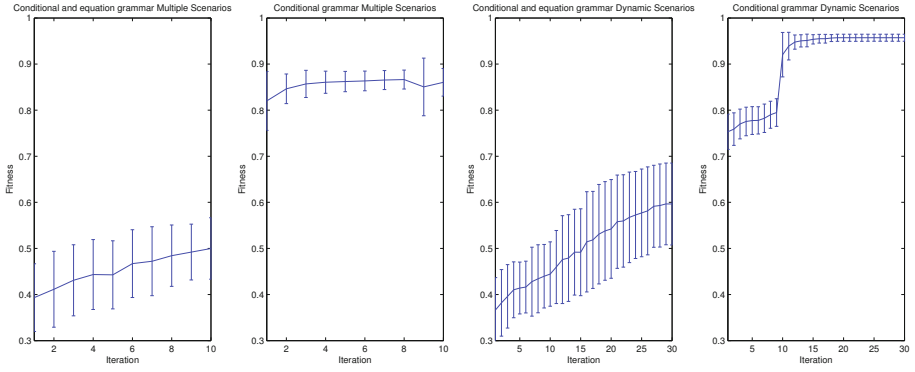
Fig. 4. In Fig. 4(a) the average validation fitness of the first front is shown, which is slightly increasing. The values decrease when more solutions with lower average are added to the first front. In Fig. 4(b) comparison of evolutionary run times for the dynamic and stationary training scenarios. On average run times are considerably lower for the dynamic training scenarios, and on comparison of the grammars adopted the SRCG form provides additional gains.

Table 2. Fitness on test data for the non-extreme solutions on the first front. The columns show total number of solutions on the fronts in the runs (Total), average fitness of the solutions on the first front (Avg Fit), standard deviation (Std), median (Med), minimum (Min) and maximum (Max).

Version	Total	Avg Fit	Std	Med	Min	Max
CG STS	59	0.467	0.033	0.471	0.374	0.521
SRCG STS	97	0.301	0.077	0.316	0.130	0.458
CG DTS	84	0.349	0.172	0.421	0.031	0.506
SRCG DTS	110	0.244	0.120	0.265	0.000	0.451

the training fitness during the runs and it is not possible to compare the values between DTS and STS since the fitness in the dynamic scenario is only for the current scenario. Thus, a validation scenario was used and in Fig. 4(a) the average validation fitness of the first front is shown, which is slightly increasing.

The non-extreme solutions from the first front for each run are evaluated on the test scenarios. The average of the fitnesses and the average of the first front is chosen in order to allow simple comparisons. This approach was chosen since there can be multiple solutions with the same fitness but different phenotypes and the out-of-sample quality of the solutions is unknown. There is a significant difference in fitness according to the non-parametric Wilcoxon rank sum test for equal medians at a 0.05-level for all values. Thus, we can conclude that for the femtocell scenarios examined here the test performance was best when using the STS setup. It is worth noting that with the SRCG and the DTS some solutions generated invalid values in the test scenarios. As expected the DTS have a higher standard deviation compared to the STS.



(a) STS (b) DTS

Fig. 5. Average training fitness of non-extreme solutions

5 Conclusions and Future work

A significant issue facing the developers of the algorithms which control the behaviour of femtocells is how best to design the algorithms to handle these unforeseen, dynamic environments. In this study we examined this issue with respect to the design of fitness functions for an evolutionary algorithm which evolves algorithms to control femtocell behaviour. More specifically we asked *"Is there a difference in the robustness of solutions (out-of-sample) based on the use of stationary versus dynamic training scenarios?"*. Given the experimental setup adopted in this study it was found that, while the dynamic training scenarios result in more efficient run times, the stationary training scenarios produce more robust solutions. In future work we will examine different approaches to the dynamic environment setup, and adopt a wider range of scenarios in each case. There are also potentially many lessons to be learned from, for example, the statistical machine learning literature on best to design training to achieve solutions which generalise beyond training data (e.g., [7, 3, 9, 2]). We will examine if these methods can complement the evolutionary search adopted here.

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

References

[1] Alba, E., Chicano, J.F.: Evolutionary algorithms in telecommunications. In: MELECON 2006, pp. 795–798. IEEE (2006)
[2] Bersano-Begey, T.F., Daida, J.M.: A discussion on generality and robustness and a framework for fitness set construction in genetic programming to promote robustness. In: Genetic Programming Conference, pp. 11–18 (1997)
[3] Breiman, L.: Bagging predictors. Machine Learning 24(2), 23–140 (1996)

[4] Byrne, J., O'Neill, M., McDermott, J., Brabazon, A.: An Analysis of the Behaviour of Mutation in Grammatical Evolution. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 14–25. Springer, Heidelberg (2010)
[5] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE TEC 6(2), 182–197 (2002)
[6] Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Springer (April 2009)
[7] Efron, B.: Bootstrap methods: another look at the jackknife. The Annals of Statistics 7(1), 1–26 (1979)
[8] Fagen, D., Vicharelli, P.A., Weitzen, J.: Automated wireless coverage optimization with controlled overlap. IEEE Transactions on Vehicular Technology 57(4), 2395–2403 (2008)
[9] Freund, Y., Schapire, R.: A Desicion-Theoretic Generalization of On-line Learning and an Application to Boosting. In: Vitányi, P. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
[10] Harper, R.: Ge, explosive grammars and the lasting legacy of bad initialisation. In: WCCI 2010 (July 2010)
[11] Hemberg, E., Ho, L., O'Neill, M., Claussen, H.: A symbolic regression approach to manage femtocell coverage using grammatical genetic programming. In: GECCO, pp. 639–646. ACM (2011)
[12] Ho, L., Ashraf, I., Claussen, H.: Evolving femtocell coverage optimization algorithms using genetic programming. In: 2009 IEEE Personal, Indoor and Mobile Radio Communications, pp. 2132–2136. IEEE (2010)
[13] Jain, R., Chiu, D.M., Hawe, W.R.: A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Eastern Research Laboratory, Digital Equipment Corp. (1984)
[14] Mckay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based Genetic Programming: a survey. Genetic Programming and Evolvable Machines 11(3), 365–396 (2010)
[15] O'Neill, M., Nicolau, M., Brabazon, A.: Dynamic environments can speed up evolution with genetic programming. In: GECCO, pp. 191–192. ACM (2011)
[16] Siomina, I., Varbrand, P.: Automated optimization of service coverage and base station antenna configuration in umts networks. IEEE Wireless Communications 13(6), 16–25 (2006)