A Novel Fireworks Algorithm with Wind Inertia Dynamics and its Application to Traffic Forecasting

Ibai Laña*, Javier Del Ser*,†,‡ and Manuel Vélez†
*TECNALIA, 48170 Derio, Bizkaia, Spain
Email: {ibai.lana, javier.delser}@tecnalia.com
†University of the Basque Country UPV/EHU, Bilbao, Bizkaia, Spain
Email: {javier.delser, manuel.velez}@ehu.eus
‡Basque Center for Applied Mathematics (BCAM), Bilbao, Bizkaia, Spain

***Abstract*—Fireworks Algorithm (FWA) is a recently contributed heuristic optimization method that has shown a promising performance in applications stemming from different domains. Improvements to the original algorithm have been designed and tested in the related literature. Nonetheless, in most of such previous works FWA has been tested with standard test functions, hence its performance when applied to real application cases has been scarcely assessed. In this manuscript a mechanism for accelerating the convergence of this meta-heuristic is proposed based on observed wind inertia dynamics (WID) among fireworks in practice. The resulting enhanced algorithm will be described algorithmically and evaluated in terms of convergence speed by means of test functions. As an additional novel contribution of this work FWA and FWA-WID are used in a practical application where such heuristics are used as wrappers for optimizing the parameters of a road traffic short-term predictive model. The exhaustive performance analysis of the FWA and FWA-ID in this practical setup has revealed that the relatively high computational complexity of this solver with respect to other heuristics makes it critical to speed up their convergence (specially in cases with a costly fitness evaluation as the one tackled in this work), observation that buttresses the utility of the proposed modifications to the naive FWA solver.**

I. Introduction

Meta-heuristics – and in particular, evolutionary algorithms – have been at the head of efficient optimization solvers over the last four decades, but it is during the last ten years when they have experienced a thriving expansion. Particularly, bio-inspired algorithms conform an increasingly copious family that grasps inspiration from the observation and emulation of assorted natural processes [1], [2].

Among these, Fireworks Algorithm (FWA, [3]) is a young heuristic optimization algorithm relying on the explosion and sparks expansion phenomena observed in fireworks. In FWA a set of n fireworks is generated, representing initial solutions drawn uniformly at random from the search space spanned by the vector of optimization variables $\mathbf{x}$ and the fitness function $f(\mathbf{x})$. Each of such fireworks *explodes*, generating stochastic sparks that also represent solutions, located in the vicinity of the originating firework, and with ranges of exploration or amplitudes that depend on the both the fitness of the firework and the parametrization of the heuristic (see Figure 1). The original implementation of the algorithm includes also a subset of Gaussian distributed sparks to part of the fireworks that adds diversity to the entire set of explored solutions. A n-sized pool of selected solutions, including the best and the worst fitness for diversity purposes, constitutes the set of fireworks (solutions) held for the next generation of the algorithm. Whereas the amplitude parameter allows balancing the trade-off between exploration and exploitation during the FWA search process, to obtain the n individuals of the next generation a total of $n + ns + m$ evaluations are needed, being ns the total number of regular sparks and m the number of gaussian distributed sparks. This increases the runtime in comparison to other optimization algorithms, specially those based on Swarm Intelligence.

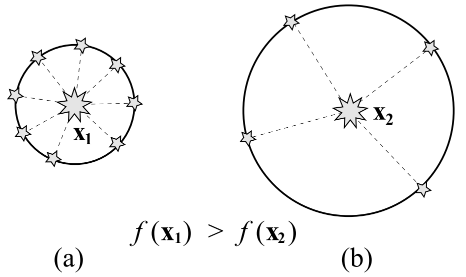

Fig. 1. Best fitness fireworks (a) explode by generating more sparks in a smaller amplitude – promoting exploitation – than worst fireworks (b), whose sparks have larger amplitude and less cardinality, fostering exploration. A maximization problem is implicitly assumed.

In this context research efforts have been lately conducted towards overcoming this issue by reducing the convergence time and consequently, improving the performance of the algorithm using as fewer generations as possible. The so-called Enhanced Fireworks Algorithm (EFWA, [4]) proposes to vary the definition of the FWA operators to address performance decay when the algorithm is applied to functions that do not have their optimum at the origin. Likewise, the authors in [5] strive to accelerate the convergence process by selecting an elite of individuals at each generation obtaining, under certain circumstances, better results than the baseline FWA. Another strategy to speed up the algorithm is parallelization, that has been performed using GPU processing [6], [7]. Other modifications to the initial FWA and to the later contributed EFWA include adaptive amplitude calculation [8], cooperation among fireworks [9] or dynamic adjusting of parameters to balance exploitation and exploration [10]. Aside from performance improvements, applications of fireworks algorithm

978-1-5090-4601-0/17/$31.00 ©2017 IEEE

to real optimization scenarios are yet scant [11], [12], [13], [14], [15] due to the relative youth of this heuristic. A practical application of interest within the scope of this work is [16], where FWA is used to optimize the parameters of a spam detection algorithm. Model parametrization via heuristic wrapper is indeed a common usage of optimization algorithms [17], using the evaluation score of the model as the fitness function of the wrapping solver.

This paper proposes a modification of the EFWA approach inspired by the physical displacement effect of fireworks due to blowing wind. We advocate that wind inertia resulting from gradients in the explored solution space is able to boost the convergence of the algorithm towards optimal solution in less iterations. Enhanced and wind inertia dynamics (WID) versions of the algorithm are tested on benchmark functions and on a parameter optimization problem, consisting of a supervised learning model aimed at predicting vehicular traffic in the short term. Statistically significant performance gains at no computational cost are shown to be achieved by our proposed scheme with respect to its baseline EFWA counterpart, both over the test functions and the practical application.

The paper is organized as follows: the proposed algorithmic changes introduced to the EFWA algorithm are detailed in Section II, whereas Section III discusses on the performance attained by both EFWA and EFWA-WID on test functions. Section IV elaborates on short-term traffic forecasting problem and the results obtained therein by the aforementioned heuristics. Finally Section V summarizes the conclusions and future research lines derived from this work.

II. Proposed FWA with Wind Inertia Dynamics

As wind inertia version of the fireworks algorithm is built on the basis of the enhanced fireworks algorithm [4], the latter introducing five essential changes with respect to its seminal version: 1) a minimal explosion amplitude check; 2) an operator that displaces randomly some of the explosion sparks; 3) a mapping operator that reallocates those sparks generated outside the search space; 4) a displacement operator for Gaussian sparks; and 5) a selection operator to reduce runtime. EFWA requires multiple input parameters that define the number of fireworks (n), the amplitude of explosion of a firework (A_i), the number of explosion sparks per firework (ns), the dimensions of the problem (d), the number of Gaussian sparks (m) and control parameters for the minimum and maximum sparks per firework (a and b), for the maximum amplitude (A), and for the boundaries or range of exploration for each variable characterizing the problem at hand. With those parameters and the foretold changes, EFWA has been proven to outperform considerably the original FWA in terms of runtime. Our proposed version of the algorithm aims at attaining a steeper convergence curve, which finds its rationale in problems defined by costly fitness functions (as in e.g. computational simulation). In such cases less iterations are desirable, if not for the optimal solution, at least for a suboptimal one obtained faster than those with milder convergence.

To achieve this, our proposal is inspired in the effect of wind in fireworks, displacing them unidirectionally towards regions indicated by the gradient in the atmospheric pressure. Our figurative wind blows always in the direction of the best spark of an iteration, pushing the rest of sparks that will conform the fireworks of the next generation in that direction. This sacrifices exploratory capabilities of the algorithm, but this counter-effect is compensated by including the worst performing spark within the fireworks held for the next generation.

Algorithm 1: Proposed EFWA-WID Algorithm

```
Input : Fitness function $f(\mathbf{x})$, number of fireworks $n$,
        number of regular sparks $ns$, number of
        Gaussian sparks $m$, minimum and maximum
        sparks per firework $a$ and $b$, maximum
        amplitude $A$, wind speed $ws$, $maxEpochs$
Output: Optimal firework $\mathbf{x}^* \doteq \{x_j^*\}_{j=1}^d$
1  Initialize $\mathbf{X}$ (population) with $n$ random fireworks
     $\{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$; compute their fitness; set $i = 0$
2  while $i < maxEpochs$ do
3      foreach $k$ in $\{1, \ldots, n\}$ do
4          Compute $A_k$ based on $ns$ as per [4, Eq. (1)]
5          Compute $ns_k$ based on $ns$ as per [4, Eq. (2)]
6          foreach $j$ in $ns_k$ do
7              Initialize spark $\mathbf{s}^j = \mathbf{x}^k$
8              foreach dimension in $\mathbf{s}^j$ do
9                  Calculate $\Delta x = A \cdot \text{rand}(-1, 1)$
10                 Apply $\Delta x$ to random sparks
11                 If needed, map $\mathbf{s}^j$ to the potential space
12             end
13         end
14     end
15     Add $m$ Gaussian sparks as per [4, Alg. 2]
16     Obtain best performing spark $\mathbf{s}^{*,i}$ and firework $\mathbf{x}^{*,i}$
17     Obtain worst performing spark $\mathbf{s}^{\times,i}$
18     $\boxed{\text{Calculate wind inertia } \mathbf{c}^i \text{ as per (1)}}$
19     Create empty set $\mathbf{X}^{\emptyset}$; set $\mathbf{x}^{\emptyset,1} = \mathbf{s}^{*,i}$, $\mathbf{x}^{\emptyset,2} = \mathbf{s}^{\times,i}$
20     Calculate epoch coefficient $ec$
21     $\boxed{\text{Generate non-uniform sparks } \mathbf{S}^{\triangleright} \doteq \{\mathbf{s}^{\triangleright,k}\}_{k=1}^n}$
22     for $k \leftarrow 2$ to $n$ do
23         for $j \leftarrow 1$ to $d$ do
24             $\boxed{x_j^{\emptyset,k} = s_j^{\triangleright,k-1} + (c_j^i \cdot ws \cdot ec)}$
25         end
26         $\boxed{\text{Calculate fitness for } \mathbf{x}^{\emptyset,k}}$
27     end
28     $\mathbf{X} = \mathbf{X}^{\emptyset}$, $i = i + 1$
29 end
30 The output $\mathbf{x}^*$ is given by the best firework in $\mathbf{X}$
```

As represented in Figure 2, wind inertia is obtained from the sparks $\{\mathbf{x}_j^{*,i}\}_{j=1}^{ns_*}$ of the best performing firework $\mathbf{x}^{*,i}$ at epoch i as

$$\mathbf{c}^i \doteq \frac{\sum_{j=1}^{ns_*} \mathbf{x}_j^{*,i} f(\mathbf{x}_j^{*,i})}{\sum_{j=1}^{ns_*} \mathbf{x}_j^{*,i}}, \tag{1}$$

where $\mathbf{c}_i$ is a convergence vector with scalar displacements for the rest of sparks; the j-th spark $\mathbf{x}_j^{*,i}$ is a vector with d dimensions; and ns_* is the number of sparks associated to the best performing firework. Each component of the wind inertia $\mathbf{c}_i$ is a scalar displacement computed independently from each other and obtained at every iteration of EFWA-WID once all fireworks and sparks have been evaluated. When selecting the new fireworks for the next iteration, each of the values of the convergence vector are added to each dimension of selected spark. In order to modulate the intensity of the wind inertia effect, a wind strength parameter $ws \in \mathbb{R}^+$ is used as a coefficient for each convergence scalar value.

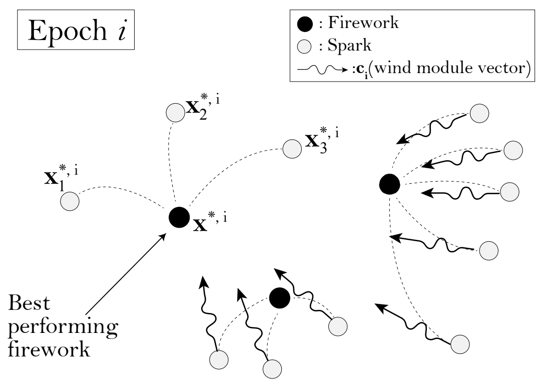

Fig. 2. Schematic representation of the dynamic wind effect at epoch i. Sparks are pushed towards the fitness-weighted center of mass of sparks generated by the best performing firework in the iteration.

During its first iterations the algorithm produces fireworks characterized by worse fitness evaluations. Consequently a convergence of all sparks to those solutions is less desirable and should be avoided. To this end, an epoch coefficient $ec \in \mathbb{R}[0, 1]$ is further introduced in the algorithm to make wind push sparks farther towards better regions in the solution space. To this end the value of ec starts with values close to 0 in early phases of the heuristic search, and is forced to increase along iterations. Furthermore, for a faster convergence, a fitness-proportionate firework generation scheme is used, similar to what is done in the widely known roulette wheel selection strategy in genetic algorithms. To do so, a vector of probabilities is computed for all the sparks, assigning higher probability of remaining in the next generation to those fireworks with better fitness value. This fitness-proportionate selection is applied independently of the wind modifications.

Algorithm 1 summarizes the fundamentals of the proposed EFWA-WID algorithm, with changes highlighted in boxes. After the initial extraction of fireworks and its sparks, wind inertia is computed based on the best performing firework. Then, the population for the next generation is generated, with the best and the worst sparks of current iteration, and an elite of random sparks pushed by the action of wind.

III. Evaluation on Test Functions

In order to gauge the impact of the proposed changes on the convergence of the solver, computer experiments with EFWA and EFWA-WID have been performed on a benchmark comprising six standard test functions for single-objective optimization: Ackley, Cigar, Sphere, Rosenbrock, Rastrigin, and Griewank [18]. To ensure a fair comparison, the parameters have been configured identically for all the tests and for both versions of the algorithm. In particular problem instances of $d = 10$ dimensions will be considered, with $n = 10$ fireworks kept among iterations, a reference value of $ns = 10$ regular sparks (the number of sparks ns_f for each firework is varied in runtime as per [4, Eq. (2)]), and $m = 5$ Gaussian sparks. Control parameters for minimum and maximum number of sparks per firework are set to $a = 0.04$ and $b = 0.8$, while the maximum amplitude parameter is $A = 80$. For each test function, both algorithms perform $maxEpochs = 5000$ iterations, which is deliberately set two or three orders of magnitude lower than previous works [3], [4] since the inclusion of wind inertia is aimed at speeding up the convergence in the early stages of the algorithm. Experiments are run afresh for a total of 20 Monte Carlo experiments, which will allow shedding light on the statistical behavior of the algorithms in the benchmark. After a fine-grained grid search for the best parameter configuration, wind speed parameter (ws) is set to values 0.1, 0.5 and 0.9, making the process consist of 4 executions (one without and three with wind) of 20 Monte Carlo experiments of 5000 iterations; algorithm solving function is therefore run 400000 times, each of them including between 50 and 100 fitness evaluations (depending on the number of sparks generated). Once configured, all tests have been run on a Linux implementation deployed in an Intel Xeon machine @ 3.07 GHz, yielding a running time of about 3 hours for the entire benchmark.

TABLE I
Averaged fitness difference at different epochs and at the epoch with highest relative performance difference

Function	**Average fitness at different epochs**			
	Max. difference (epoch)	Epoch		
		500	100	50
Ackley	65.5% (984)	58.6%	24.4%	18.8%
Cigar	81.61% (433)	80.7%	72.4%	69.6%
Sphere	82.9% (538)	80.5%	70.3%	66.1%
Rosenbrock	65.4% (23)	40.1%	47.9%	60.6%
Rastrigin	73.2% (4969)	42.2%	25.4%	18.1%
Griewank	77.3% (478)	62.9%	68.1%	61.2%
Average	74.3%	62.98%	51.4%	49.1%

The discussion begins in Figure 3, where convergence plots are shown with different scales for a better visual performance assessment. In all plots it can be noted that in early stages of the search process wind inertia entails a faster convergence than the baseline EFWA. However, for most functions wind is more effective when applied with less intensity as driven by the value of ws. A higher wind speed produces results closer to those of the original EFWA, except for the Ackley function for which there are no significant differences among different wind speeds. Considering the scale of each function, all four versions reach similar results after the whole run, although this end will be addressed afterwards.

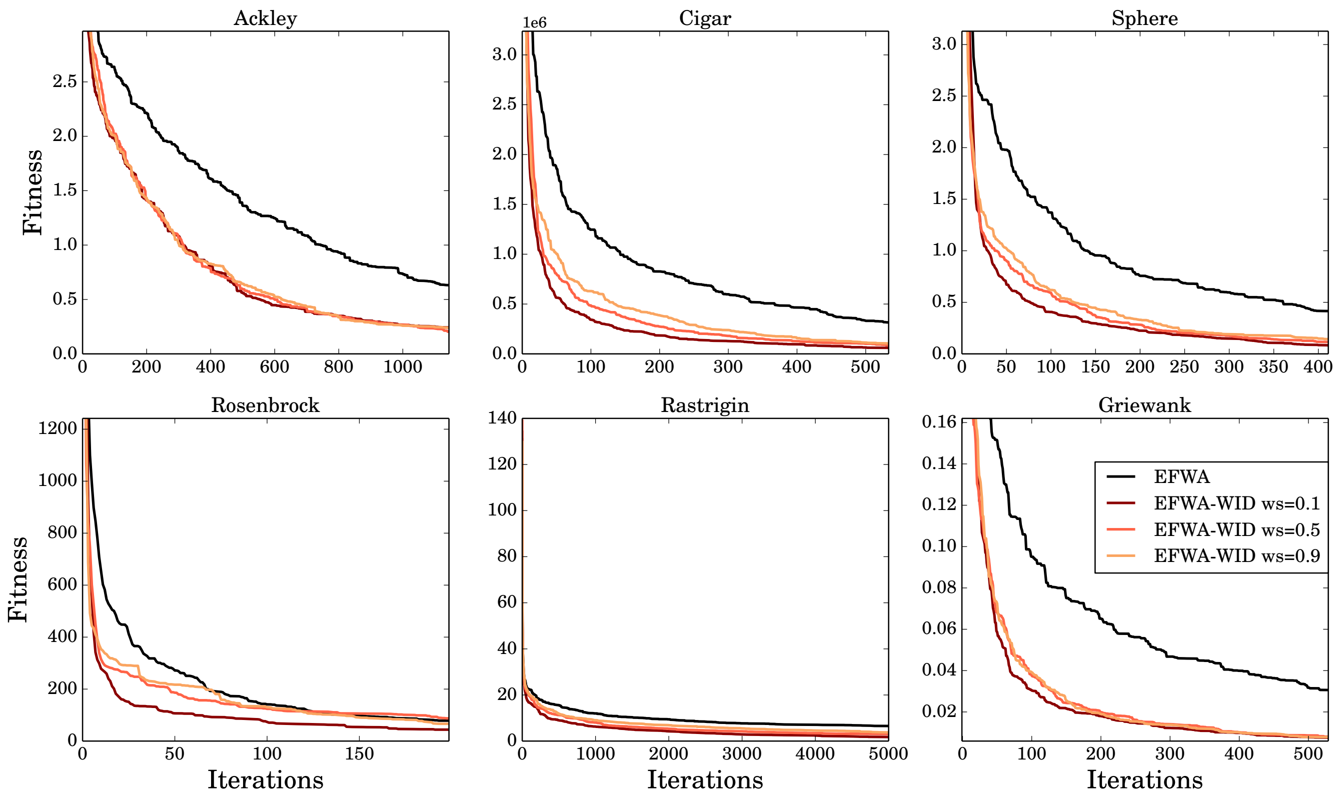

Fig. 3. Convergence of the average fitness value along iterations for the EFWA and EFWA-WID heuristics during the initial iterations where differences were found to be higher. In all cases the introduction of wind inertia accelerates the convergence of the algorithm without degrading – with statistical significance, see Table II – the finally optimized value of the objective.

In light of the performance improvement observed in these plots we next inspect the evolution of the relative difference between results obtained with each version of the algorithm. In this regard Table I collects the relative performance increase between EFWA and EFWA-WID with $ws = 0.1$ at different epochs of the search process, including the generation that scored the highest relative performance increase. At this point of maximum discrepancy an average of 74.3% relative improvement is found for EFWA-WID over EFWA when averaged over all test functions, although the epoch at which such a performance improvement can be expected varies significantly among test functions. Also remarkable is the fact that with wind inertia, four of the functions achieve a relative improvement higher than 60% by epoch 50, a relevant aspect when the target is to speed up the algorithm due to e.g. a computationally heavy fitness function.

Figure 4 depicts the evolution of the performance gain of EFWA-WID (with $ws = 0.1$) over EFWA through epochs for all test functions. While EFWA-WID features a constant performance gain along iterations for the Rastrigin function, in the rest of test functions the gain is positive during the first 1000 epochs and degrades thereafter. As a result of smaller scales, little changes produce acuter increments (or decrements) in the last epochs.

In order to assess the statistical significance of these differences among versions of the algorithm, Wilcoxon rank-sum tests have been performed over samples of 20 individual fitness values at different epochs to assess whether the medians of their populations differ without assuming that such values

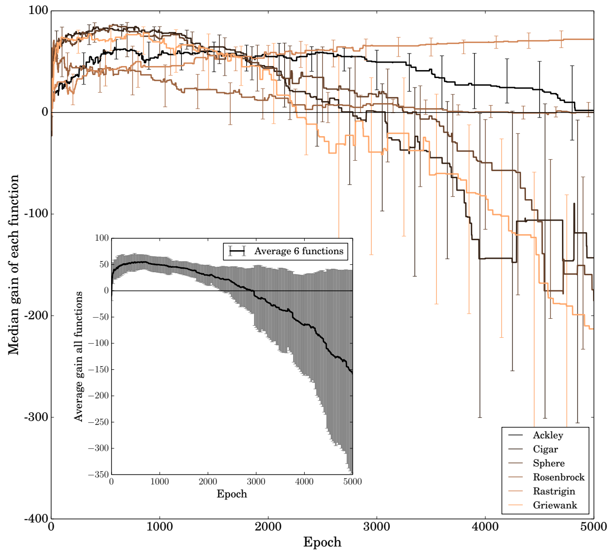

Fig. 4. Evolution curves of the performance gain between EFWA and EFWA-ID with $ws = 0.1$. The inner plot depicts median and interquartile range computed over the aforementioned curves.

are normally distributed. Table II presents p-values obtained from these tests, from which it can be concluded that the performance gaps are always statistically relevant in the epochs where such a gap scores higher.

TABLE II
WILCOXON P-VALUE COMPUTED BETWEEN RESULT SETS OF EFWA AND EFWA-WID WITH $ws \in \{0.1, 0.5, 0.9\}$

Function	**Wilcoxon p-value at epoch 5000**		
	$ws = 0.1$	$ws = 0.5$	$ws = 0.9$
Ackley	0.29	0.08	0.17
Cigar	$< 10^{-3}$	0.005	$< 10^{-3}$
Sphere	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Rosenbrock	0.29	0.62	0.70
Rastrigin	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Griewank	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Function @ epoch	**Wilcoxon p-value at epoch with max. difference**		
	$ws = 0.1$	$ws = 0.5$	$ws = 0.9$
Ackley @ 632	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Cigar @ 31	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Sphere @ 73	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Rosenbrock @ 24	$< 10^{-3}$	0.0168	$< 10^{-3}$
Rastrigin @ 826	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$
Griewank @ 84	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$

IV. EVALUATION ON REAL DATA: TRAFFIC FORECASTING

Forecasting traffic conditions is a relevant functionality of Intelligent Transport Systems (ITS), as it provides decision makers with essential information of utmost utility for traffic regulation and planning. Predicting variables like traffic volume, road occupancy, level of service or average speed has been addressed with diverse techniques since the early 80s, from time-series analysis to non-parametric and machine learning methods such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), Bayesian Networks or Fuzzy Logic models [19]. Traffic-related data have become increasingly available around the world along with the advent and proliferation of open data initiatives. This has ignited the pace at which advances in data-driven techniques have lead to more precise predictions over wider geographical areas [20]. Machine learning methods are frequently chosen for these tasks, often with relative success over naïve (historic average and last measurement predictions) and time-series models. However, many of these algorithms operate in a black-box manner, with their parameter setting often fine-tuned by means of a trial-and-error procedure.

Evolutionary algorithms are a powerful tool to automate the tuning of traffic forecasting machine learning methods operating as a wrapping algorithm driven by the performance of the wrapped predictive model. This approach has been explored in the literature mainly with Genetic Algorithms (GA) [21], [22], yet recently with alternative bio-inspired methods such as Particle Swarm Optimization (PSO) [23], [24]. In this paper both EFWA and EFWA-WID will be applied to obtain the optimal parameter configuration of a random forest regressor [25], an extensively used supervised learning model in forecasting applications built upon the iterated randomized sampling of the dataset and the construction of ensembles composed by weak tree learners as an effective means to implement bagging and avoid overfitting.

A. Data Composition and Model Building

Data to build the regression model are obtained from a public data repository [26] comprising traffic flow records of a magnetic loop sensor placed in a center area of Madrid (Spain). The location of the sensor under study has been selected according to a completeness criterion, as none of the loops deployed in this city has fully complete and valid data. The selected loop is one of the devices with most complete data in the repository, with ca. 30000 valid flow readings for 2015. With these data and their timestamps datasets are built based on three specifications: 1) step, i.e. the time between two readings (in this case 15 minutes); 2) depth or window size, namely, the time span of past observations that are used in each instance; and 3) prediction horizon, i.e. the number of steps into the future for which the prediction is made. At the time of the experiments, only three months of data were available for 2016, which have been used for model testing. This being so, the training and validation dataset covers the first three months of 2015, as depicted in Figure 5. Using the same time frame for training (2015) and testing (2016) minimizes potential byproducts in the predictive scores due to the existence of seasonalities.

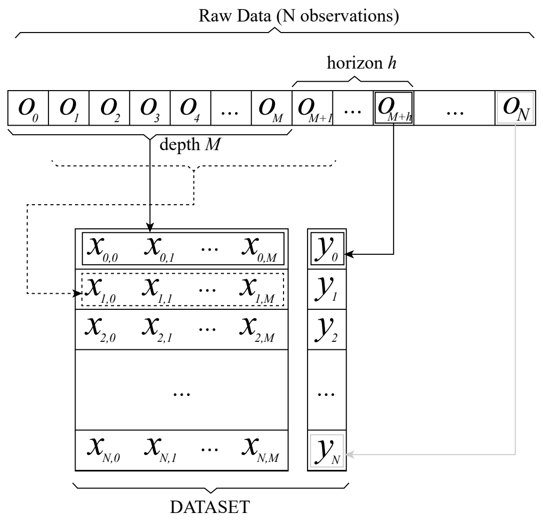
Raw Data (N observations)
horizon h
O_0 O_1 O_2 O_3 O_4 ... O_M O_{M+1} ... O_{M+h} ... O_N
depth M
$x_{0,0}$ $x_{0,1}$ ... $x_{0,M}$
$x_{1,0}$ $x_{1,1}$ ... $x_{1,M}$
$x_{2,0}$ $x_{2,1}$ ... $x_{2,M}$
$x_{N,0}$ $x_{N,1}$ ... $x_{N,M}$
y_0 y_1 y_2 ... y_N
DATASET

Fig. 5. Dataset building process.

Given a timestamp flow readings form samples of a dataset with M features (corresponding to the window size), which represent the M observations prior to the timestamp. The y^{th} slot after the timestamp defines the target variable. Forecast horizon is set to 4 steps (i.e. 1 hour into the future), which aligns with other short-term traffic prediction horizons found in the literature [19], [20]. Initially, predictions are based on observations from the previous 2 hours, so the window size parameter has a default value of $M = 8$. This initial window size is an arbitrary one, as optimality of this value depends deeply on the characteristics of the sensor location. Furthermore, despite the low value of M selected as the window size it should be noted that our goal is to show that even in a supervised learning problem with minor needs for feature selection the selection of one heuristic or another as a configuration wrapper does make a difference in terms of predictive accuracy. Thus, tailoring window size is one of the aims of EFWA and EFWA-WID optimization in this research. Many other evolutionary techniques have been adopted earlier with this purpose [27], but to the best of authors' knowledge, none of them deals with fireworks algorithm nor with improved variants of this solver as the one proposed in this work.

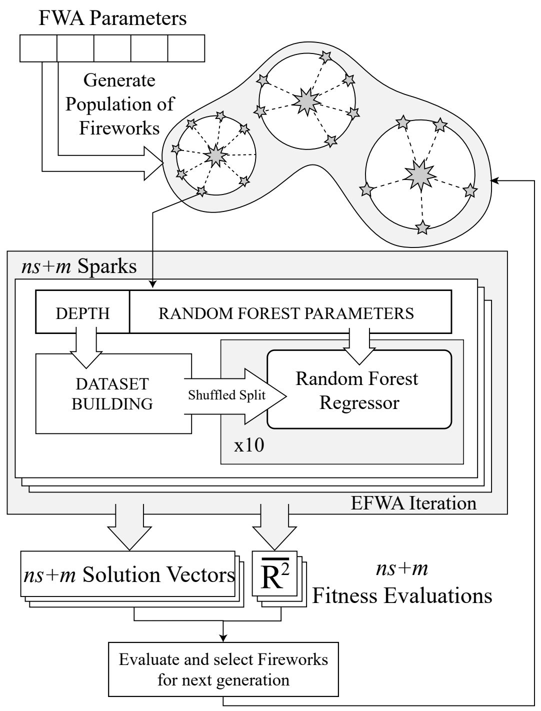

Fig. 6. Integration of dataset building and regressor in the firework wrapper.

Both EFWA and EFWA-WID algorithms are used as a wrapper that seeks 4-dimensional solutions. The first dimension is the optimal window size of the dataset, whereas the remaining three dimensions represent configuration parameters of the Random Forest Regressor model: number of estimators, i.e. the amount of trees in the forest; the maximum depth of trees; and the maximum amount of leaf nodes. Each time a solution (spark or firework) obtained in EFWA or EFWA-WID is evaluated to obtain its fitness, a new dataset is generated due to the fact that a variation of window size parameter unchains different dimensions for the training and test instances. To obtain the output of the random forest regressor, the dataset is randomly shuffled and 10-fold cross-validated, executing the algorithm 10 times and obtaining 10 R^2 metrics, which averaged give rise to the fitness of the model. The wrapping mechanism is schematically depicted in Figure 6.

B. Results and Discussion

Each evaluation takes between 30 seconds and 2 minutes in an Intel Xeon processor, depending on the size of the dataset and the random forest parameters. As foretold in previous section, one epoch of FWA can easily reach 100 fitness function evaluations, rendering optimization a highly time demanding problem. Thus, in order to test the performance of both EFWA and EFWA-WID we have configured the algorithms to execute a total of 40 epochs, a substantially lower amount of iterations than usual, expecting that the improvements proposed in this paper allow an earlier convergence. Our enhanced FWA-WID has shown that with some datasets, a good fitness solution can be obtained in less than 100 epochs. Even after reducing significantly the number of algorithm generations, running 10 Monte Carlo experiments with traffic data, a dynamic dataset building (due to the consideration of the time window as an optimization variable) and a random forest predictive with 10-fold cross-validation has taken more than 20 days on two dedicated machines (one for each algorithm).

The rest of FWA parameters are set to the values used with test functions: $n = 10$, $ns = 10$, $m = 5$, $a = 0.04$, $b = 0.8$, $A = 80$, and $ws = 0.1$ in EFWA-WID version, as it is the best performing of the ones tested before. In addition, all variables considered for the traffic prediction problem are discrete, which requires an adaptation of FWA as it usually operates in a continuous space. Random generation of values for solution vectors is made with integer random generators, whereas real values resulting displacements in the solution space (e.g. wind) are rounded to its nearest integer value. This leads to a more coarse-grained set of solutions, and results are more staggered through epochs.

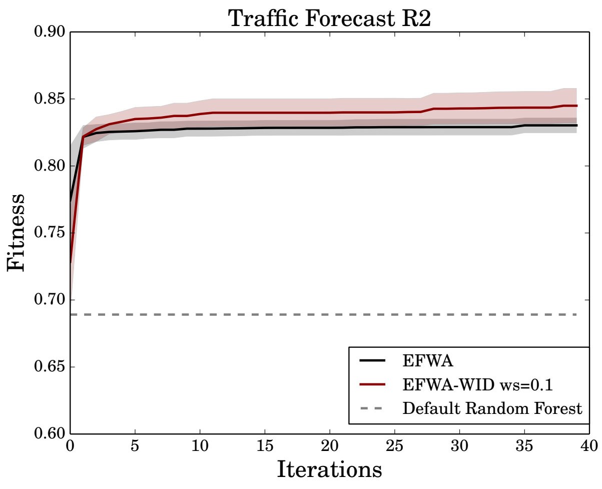

Fig. 7. Average R^2 performance (bold lines with shaded region bounded by the standard deviation of results over 10 Monte Carlo experiments) of EFWA, EFWA-WID and a non-optimized default regression model over 40 iterations.

Figure 7 shows the evolution of the R^2 score with both versions of the algorithm versus that obtained with an off-the-shelf configuration for the model (10 trees, no maximum depth, no maximum leaf nodes) and a 4-step window size. Both EFWA and EFWA-WID outperform easily the random forest model configured with default parameter values starting as early as in the second iteration. A significant 18% increase is observable for EFWA (from an average R^2 = 0.68 to

$R^2 = 0.83$), while EFWA-WID starts at a lower initial fitness, converging fast to obtain superior results from epoch 2 onwards. After 30 epochs, both algorithms stop to converge to their final score results: a R^2 coefficient of 0.845 for EFWA-WID and 0.83 for EFWA. Although a deeper test could eventually render better results, these performances represent good estimators of one-hour-ahead traffic. However, the wind powered version of the algorithm improvement is not as acute as for the test functions.

The rationale for this performance gap degradation might reside in the discretization of the solution space: within the exploration range of defined fireworks, a vocabulary of less than 100 possible values are defined for each dimension of the solution vector. This low granularity and the operators of the heuristic solver makes it specially critical to properly fine-tune the parameters of the optimization algorithm, and increases the chance to get stuck in local optima if this model tuning is not performed. A grid search of the optimum parameters with the same boundaries would have taken ca. 100^4 evaluations of the R^2 score for each algorithm, by any means a higher amount than the number of evaluations done in the above performance plots. Anyhow, results obtained in first generations outline good performance of FWA for this practical application, although a good definition of the algorithm execution parameters (more fireworks, less iterations) might be a required a priori condition for every simulated solver.

V. Conclusions and Future Work

In this paper we have presented a modification of the enhanced fireworks algorithm EFWA inspired in wind inertia dynamics, which has been tested, along with its original counterpart, over six optimization test functions, obtaining for all cases an earlier convergence to optimal solution. Being FWA a relatively young algorithm for which examples of real applications are scarce, a second contribution of this work has been to test its enhanced and wind inertia versions on a traffic flow prediction problem, jointly obtaining the optimal regression parameters and dataset configuration. Retrieving the fitness evaluation of this kind of problem is computationally expensive, for which each epoch of firework algorithms, with several evaluations involved, can take a considerable amount of time. This requires adaptations in the algorithm for this class of high computation problems, and any improvement for an earlier convergence like the wind inertia proposed in this paper, might be of utmost utility. In practice, we have observed a huge computation time demand and a very early convergence to a near optimal solution with both algorithms, obtaining a slight performance difference between their final results.

Other FWA research initiatives point in the direction of parallel computing, which combined with early convergence strategies can render the algorithm even more effective. By undertaking this research line a conveniently adapted FWA might also become a good choice for optimization of problems with high computational demands. Indeed this postulated hypothesis will steer future research efforts towards implementing this algorithm in Big Data architectures, jointly leveraging the availability of computation cores and efficient programming functionalities for large datasets for a proper scalability of this search heuristic.

Acknowledgements

This research work has been partially funded by the Seventh Framework Programme (FP7/2007-2013) of the European Commission under Grant Agreement 608885, and by the Basque Government under the ELKARTEK programme (BID3ABI project).

References

[1] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *Elektrotehniški Vestnik*, 2013.

[2] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1980–1987.

[3] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International Conference in Swarm Intelligence*. Springer, 2010, pp. 355–364.

[4] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 2069–2077.

[5] Y. Pei, S. Zheng, Y. Tan, and H. Takagi, "An empirical study on influence of approximation approaches on enhancing fireworks algorithm," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2012, pp. 1322–1327.

[6] K. Ding, S. Zheng, and Y. Tan, "A gpu-based parallel fireworks algorithm for optimization," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 9–16.

[7] Y. Tan, "Implementation of fireworks algorithm based on gpu," in *Fireworks Algorithm*. Springer, 2015, pp. 227–243.

[8] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3214–3221.

[9] Y. Tan, "Cooperative fireworks algorithm," in *Fireworks Algorithm*. Springer, 2015, pp. 133–149.

[10] J. Liu, S. Zheng, and Y. Tan, "The improvement on controlling exploration and exploitation of firework algorithm," in *International Conference in Swarm Intelligence*. Springer, 2013, pp. 11–23.

[11] A. Janecek and Y. Tan, "Iterative improvement of the multiplicative update nmf algorithm using nature-inspired optimization," in *Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 3. IEEE, 2011, pp. 1668–1672.

[12] H. Gao and M. Diao, "Cultural firework algorithm and its application for digital filters design," *International Journal of Modelling, Identification and Control*, vol. 14, no. 4, pp. 324–331, 2011.

[13] L. Bing, Y. Ning, and D. YeHong, "Lav path planning by enhanced fireworks algorithm on prior knowledge," *Applied Mathematics and Nonlinear Sciences*, vol. 1, no. 1, pp. 63–73, 2016.

[14] A. M. Imran and M. Kowsalya, "A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 62, pp. 312–322, 2014.

[15] R. Rajaram, K. Palanisamy, S. Ramasamy, and P. Ramanathan, "Selective harmonic elimination in pwm inverter using fire fly and fire works algorithm," *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, vol. 1, pp. 55–62, 2014.

[16] W. He, G. Mi, and Y. Tan, "Parameter optimization of local-concentration model for spam detection by using fireworks algorithm," in *International Conference in Swarm Intelligence*. Springer, 2013, pp. 439–450.

[17] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[18] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[19] C. Van Hinsbergen, J. Van Lint, and F. Sanders, "Short term traffic prediction models," in *Proceedings of the 14th World Congress on Itelligent Transport Systems (ITS)*, 2007.

[20] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where were going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[21] ——, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 3, pp. 211–234, 2005.

[22] P. Lopez-Garcia, E. Onieva, E. Osaba, A. D. Masegosa, and A. Perallos, "A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 557–569, 2016.

[23] A. Nagare and S. Bhatia, "Traffic flow control using neural network," *International Journal of Applied Information Systems*, vol. 1, no. 2, pp. 50–52, 2012.

[24] S.-y. Liu, D.-w. Li, Y.-g. Xi, and Q.-f. Tang, "A short-term traffic flow forecasting method and its applications," *Journal of Shanghai Jiaotong University (Science)*, vol. 20, pp. 156–163, 2015.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] Open data portal of the madrid city council. [Online]. Available: http://datos.madrid.es/portal/site/egob

[27] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Spatio-temporal short-term urban traffic volume forecasting using genetically optimized modular networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 5, pp. 317–325, 2007.