International Journal of Approximate Reasoning 121 (2020) 23–38

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

An Ant Colony Optimization approach for symbolic regression using Straight Line Programs. Application to energy consumption modelling

R. Rueda [a,b,*], L.G.B. Ruiz [a,b], M.P. Cuéllar [a,b], M.C. Pegalajar [a,b]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Spain
[b] C/. Pdta. Daniel Saucedo Aranda s.n., 18071 Granada, Spain

ARTICLE INFO

Article history:
Received 18 June 2019
Received in revised form 9 March 2020
Accepted 16 March 2020
Available online 18 March 2020

Keywords:
Energy efficiency
Symbolic regression
Straight Line Programs
Ant Colony Optimization

ABSTRACT

The increase of energy consumption and their direct effects on pollution and global warming have motivated governments to develop new strategies to promote a better usage of energy. One of the most important aspects related to energy efficiency is the need for a suitable model of energy consumption that can be used to make predictions or to aid experts in high level decision making processes. Symbolic regression techniques can be used to discover an energy consumption model that fits these purposes. Traditionally, the problem of symbolic regression has been solved by using genetic programming approaches to find the algebraic expression that best fits the regression problem data, where each expression is encoded as a tree structure. In previous works, we found that a different approach using Straight Line Programs as a representation technique could provide promising results for symbolic regression, although the size of the resulting algebraic expression might be increased when compared to the traditional approach. This work proposes an Ant Colony Optimization algorithm for Straight Line Programs to solve the problem, and makes a study to compare the approach with traditional genetic programming in a real energy consumption modelling problem.

1. Introduction

The increase of energy consumption in the building sector has become a major problem for many governments in the developed world, due to limited energy sources, increases in the price of energy and production costs, and high emissions of CO_2. For all of the aforementioned reasons, the research efforts to reduce energy consumption and to use energy efficiently have been increased substantially during the past two decades [1]. More specifically, the advances in sensor technologies and communications allow us to study multiple problems regarding energy efficiency research, such as energy consumption forecasting [2,3], anomaly detection [4,5], consumer profile mining [6,7], energy demand planning [8], and energy consumption modelling [9], among others.

In our research, we are interested in the problem of energy consumption modelling, whose objective is to find mathematical or computational models that help to accurately approximate, or explain, energy consumption behavior. Examples of approaches to model energy consumption in the literature are reference [10], which uses a Bayesian semi-parametric quan-

* Corresponding author.
E-mail addresses: ramonrd@ugr.es (R. Rueda), bacaruiz@ugr.es (L.G.B. Ruiz), manupc@decsai.ugr.es (M.P. Cuéllar), mcarmen@decsai.ugr.es (M.C. Pegalajar).

https://doi.org/10.1016/j.ijar.2020.03.005
0888-613X/

tile regression technique to model the energy consumption in a municipal wastewater plant; the study of computational intelligence methods to model the household electricity consumption in [11], the use of intelligent techniques (Genetic Programming, Multiple Regression, Artificial Neural Network, etc.) to build models that estimate the energy consumption of a building using weekdays energy consumption and outdoor data (temperature, wind speed, humidity, etc.) in [12], and reference [13], which makes use of machine learning techniques (linear regression, boosting, SVM, etc.) to estimate hourly energy consumption in residential buildings. As we can see, energy consumption modelling has been applied mostly to solve forecasting and correlation discovery problems, although it can also be applied in other scopes such as anomaly detection. Examples of previous efforts in this topic are reference [14], which describes a method of detecting abnormal energy consumption in buildings using machine learning, or reference [15], which proposes a ensemble anomaly detection framework that helps the building manager in decision-making problems.

Designing a suitable model of energy consumption depends greatly on the formulation of the problem and the requisites to be accomplished in the solution. However, although several works have emerged that accurately solve modelling or prediction problems, they fall in its low interpretability [16]. Consequently, recent works attempt to find accurate and interpretable models. By means of example, this is the case of the proposal in reference [17], which developed a method based on decision trees to forecast energy consumption in buildings; or the approach in reference [18], which proposed a genetic fuzzy system that builds interpretable knowledge bases for predicting energy consumption in smart buildings.

Therefore, as has been argued by Bratko in [19] some data mining applications need to find a balance between accuracy and interpretability, such as applications of decision making in the scope of energy efficiency research. For this reason, in this work we use symbolic regression [20] for energy consumption modelling. More specifically, the solution found by symbolic regression can be represented as algebraic expressions that can approximate the energy consumption data. Then, these algebraic expressions are sufficiently interpretable to provide an explanation of the energy consumption behavior, and to be used for high-level decision making. Just to cite some scenarios where other researchers use symbolic regression to solve energy consumption problems, we find the work [21], which uses Genetic Programming (GP) to estimate demand and energy consumption, providing more accurate results than traditional regression analysis. The study [22] uses Linear Genetic Programming to perform consumer electricity demand forecasting, and our previous work in reference [23] studies Straight Line Programs and tree representation performance for the energy consumption modelling of a set of buildings in a compound.

Despite their potential benefits, algorithms and methods of the current techniques to solve symbolic regression are still under study. Examples of these problems that directly affect our research goals are a) the *bloating* problem [24,25], which consists of the increase of size of variable-length representations during the process of symbolic regression-solving; b) the *representation* problem, aimed at finding the best symbolic representation model that helps to reduce the search space and to facilitate optimization algorithm design; or c) the *generalization* problem to prevent overfitting, which also relates to find simpler algebraic expressions with reduced size that model training data patterns accurately. Different studies were carried out to minimize these limitations [23,26] which use fixed-size structures such as Straight Line Programs (SLPs) to avoid bloating and representation problems. In the cited article, we concluded that SLPs are able to provide solutions with equal or better accuracy than neural networks in some cases, especially when the neural network models are recurrent and training algorithm gets easily trapped in local optima. In this article we propose an algorithm inspired by Ant Colony Optimization (ACO) [27] to find accurate symbolic regression solutions with reduced size with regards to Genetic Programming algorithms used in the literature [23,28].

In this work, we validate our proposal over a set of energy consumption data of public buildings at the University of Granada. We address our research by assuming that if exists a correlation between the energy consumption of the working days, then we can develop a method able to detect which days are related and how, and find an interpretable solution with high accuracy that explains the energy consumption of a working day in terms of the energy consumption of the remaining working days. The main contribution of this article is the formulation of SLP training as a graph traverse problem for its use within the ACO paradigm, and the design of algorithm components to help us to obtain symbolic regression solutions of a lower size regarding the genetic programming approach. The proposal is firstly validated over a classical ACO approach to compare the results between two approaches formulated as a graph traverse problem, and then we make a comparison with classical genetic algorithms to test the quality of the solutions found. To achieve these objectives, this manuscript is structured as follows: Section 2 describes the main concepts regarding symbolic regression and Ant Colony Optimization, as an introduction to the methods and techniques developed in this piece of research. Section 3 introduces the ACO approach. Experiments are conducted in real energy consumption data problems, and then analyzed in Section 4. Finally, Section 5 concludes and discusses future work.

2. Background in symbolic regression and Ant Colony Optimization

2.1. Symbolic regression and the representation problem

Given a set of so-called independent variables $\vec{x} = (x_1, x_2, ..., x_n)$ and dependent variables $\vec{y} = (y_1, y_2, ..., y_m)$, where $x_i, y_j \in \mathbb{R}, \forall i, j : 1 \leq i \leq n, 1 \leq j \leq m$, symbolic regression attempts to find an algebraic expression $\tilde{f}(\vec{x}, \vec{w})$ and parameters $\vec{w}$, where $\vec{w} = (w_1, w_2, ..., w_k)$, $w_i \in \mathbb{R}, \forall i : 1 \leq i \leq k$, such as $\vec{y} \approx \tilde{f}(\vec{x}, \vec{w})$. Symbolic regression can be viewed as an abstraction of traditional regression analysis techniques widely used in engineering and scientific research, such as

linear regression or logistic regression. In traditional regression analysis, the regression hypothesis f is established in advance, and the objective is to find the values for parameters $\vec{w}$ that minimize an error measurement, as for instance $e(f(\vec{x},\vec{w}),\vec{y}) = ||f(\vec{x},\vec{w}) - \vec{y}||$. On the other hand, symbolic regression assumes not only that $\vec{w}$ are unknown in advance, but also f, and the objective is to find an approximation $\tilde{f}(\vec{x},\vec{w})$ of the optimal algebraic expression that minimizes $e(\tilde{f}(\vec{x},\vec{w}),\vec{y})$.

Symbolic regression problems have traditionally been addressed from the perspective of supervised learning in the machine learning community, where $\vec{x}$ and $\vec{y}$ are the input and output data, respectively, and the goal is to perform a search over a space of algebraic expressions to find the best expression $\tilde{f}$ that minimizes $e(\tilde{f}(\vec{x},\vec{w}),\vec{y})$. Since the space of algebraic expressions is large [29,30], heuristic global search methods, such as Genetic Programming (GP) [31], have been proposed in the literature to tackle the problem. Further information about learning, representation and GP algorithm design can be found in reference [32].

The traditional representation for algebraic expressions in GP is the tree representation [31]. Recent studies in the past decade have drawn attention to alternative representations, with a special focus on linear model representations [33], due to the simplicity and potential benefits regarding the traditional non-linear representation with trees. This study highlights additional benefits of a fixed-size linear representation regarding the design of components of the optimization technique, such as the crossover and mutation operators in genetic algorithms, and the simplicity of reducing the effect of the bloating problem. Nowadays, we can find several approaches based on linear grammar representations such as Gene Expression Programming [34], Linear Programs [35], or Straight Line Programs (SLP) [36], among others.

As described in the introduction, in previous research we have explored the use of Straight Line Programs to solve energy consumption modelling problems from the perspective of Genetic Programming for symbolic regression [23,37,26], obtaining promising results regarding accuracy in real problem data. SLPs are grammar-based representations capable of encoding algebraic expressions for symbolic regression [36], and are inspired by Straight Line Grammars (SLG) [38]. SLG is a formal grammar that can be described as a tuple (V,T,P,S), where V is the set of non-terminal symbols, T is the set of terminal symbols, P is the set of production rules and S is the non-terminal starting symbol of the grammar. Each production rule in P is a context-free grammar production rule, each of these production rules cannot generate loops. A SLG in Chomsky normal form that generates a single non-empty word is a Straight Line Program. On the other hand, in the symbolic regression problem addressed in this work, the set of terminal symbols (T) is composed by a set of known mathematical operators $O \in \{o_1, o_2, ..., o_l\}$ (typically unary or binary arithmetic operators), a set of terminal input data $\{x_1, x_2, ..., x_n\}$ and a set of constant parameters $\{w_1, w_2, ..., w_k\}$. Moreover, a SLP contains N production rules $\{U_1, U_2, ..., U_N\} \in V$, where U_N is the starting symbol (S) of the grammar. Each production rule of a SLP contains a mathematical operator and two operands, whose can be a terminal symbol of T or a non-terminal symbol in V. Finally, the non-terminal symbols used in a production rule must reference subsequent production rules to avoid recursion. Then, given a SLP, the generation of the algebraic expression encoded into a SLP starts at the production rule U_N. Moreover, each non-terminal symbol U_i in the rule consequent is iteratively replaced by its associated production rule from $i = N-1$ down to $i = 1$. Formula (1) shows an example of a SLP with maximum size $N = 6$ and parameters $\vec{w} = (w_1, w_2, w_3) = (4, 8, 3)$. If we apply the described procedure, then the algebraic expression encoded can be derived from U_6 as $\tilde{f}(\vec{x},\vec{w}) = U_6$; $U_6 \Rightarrow U_3 + U_5 \Rightarrow U_3 + (U_1 + U_4) \Rightarrow U_3 + (U_1 + (w_3 + U_3)) \Rightarrow cos(U_2) + (U_1 + (w_3 + cos(U_2))) \Rightarrow cos(w_2 * x) + (U_1 + (w_3 + cos(w_2 * x))) \Rightarrow cos(w_2 * x) + (x^{w_1} + (w_3 + cos(w_2 * x)))$. For algebraic expression evaluation purposes, the parameters $\vec{w}$ should also be substituted in a last step, therefore providing the expression $\tilde{f}(x, (4, 8, 3)) = cos(8 * x) + (x^4 + (3 + cos(8 * x)))$.

$$\begin{aligned} U_1 &\rightarrow pow(x, 4) \\ U_2 &\rightarrow 8 * x \\ U_3 &\rightarrow \cos(U_2) \\ U_4 &\rightarrow 3 + U_3 \\ U_5 &\rightarrow U_1 + U_4 \\ U_6 &\rightarrow U_3 + U_5 \end{aligned} \tag{1}$$

On the other hand, additional benefits are assigned to SLPs due to it can be represented as a directed acyclic graph (DAG), which implies a potential over classical structures such as trees. For example, the study of reference [39] compares tree and graph structures regarding Genetic Programming problems, and the outcomes of this research work suggest that graph structures are a promising alternative representation regarding trees, since the graph structure allows the reuse of nodes that represent pieces of the algebraic expression and reduces the effects of the bloating problem. Nevertheless, although SLPs are fixed-size structures and the bloating problem is limited because of this representation, in previous experimentations [37] we observed that the resulting algebraic expressions obtained from SLP optimization were large with regards to their simplified form, and less interpretable. This drives the research study of this article, where we pursue the development of techniques targeted at finding a balance between SLP accuracy and size. Different methods can be found in the literature to solve this problem, such as model regularization [28], ant colony optimization [40,41], or multi-objective optimization [42], among others. As mentioned in the introduction, our proposal is inspired by ant colony optimization. Subsection 2.2 provides a background to ACO, and then Section 3 describes the approach.

2.2. Fundamentals of Ant Colony Optimization

Ant Colony Optimization [43] is a bio-inspired global search metaheuristic that belongs to the set of swarm intelligence methods [27], and it is used to solve combinatorial optimization problems defined as (S, Ω, e), where S is a search space defined over a finite set of discrete decision variables $U = \{U_1, U_2, ..., U_N\}$, Ω is a set of constraints defined over U, and $e : U_1 \times U_2 \times ... \times U_N \rightarrow \mathbb{R}_{\geq 0}$ is a loss function to be minimized. It is said that a solution $s \in S$ is feasible if all variables $U_i \in s$ have been assigned values from their domain, and they satisfy the constraints in Ω. An optimal feasible solution to the problem $s^* \in S$ verifies that $e(s^*) \leq e(s_i) \forall s_i \in S : s_i$ is feasible.

The ACO design methodology is based on the problem formulation as a graph traverse over a *construction graph* $G = (V, E)$ that represents the search space S, where V stands for the graph vertices and E for the edges. A solution $s \in S$ is incrementally built from a selected starting node of the graph. Thus, traversing the graph performs the assignment of values to variables $U_i \in s$ until the solution s is constructed. This is a simulation of the real behavior of an ant that departs from the nest to the food. Each time the ant traverses an edge of the graph (i.e., a value to a variable $U_i \in s$ has been assigned), pheromone is released to mark the edge for other members of the colony that will perform another graph traverse in the future. The literature offers a plethora of Ant Colony Optimization approaches whose algorithm components differ from each other, as for instance the way an ant chooses the path, the way pheromone is released and evaporated, parallel algorithm approaches, etc. We refer the reader to [44] for a survey on ACO methods.

The first Ant Colony Optimization method was proposed in [43], and it is known as the *Ant System* optimization. The *Ant System* used a single ant to solve the problem. Nowadays, *Ant Colony Optimization* refers to a variation of this approach where not a single ant, but a population of ants, are deployed together over the construction graph to find the best solution to the problem addressed, and uses heuristic values that encode expert information about the problem instance definition in order to speed up the search procedure. Classically, the heuristic information that defines the construction path is encoded as α and β values in a formula, as for instance Formula (2). We refer the reader to the work [45] for a more detailed explanation of the algorithm's component design.

$$p^i_{jk}(t) = \begin{cases} l \dfrac{\left[\tau_{jk}(t)\right]^\alpha \left[\nu_{jk}(t)\right]^\beta}{\sum_{l \in N^i_j(t)} \left[\tau_{jl}(t)\right]^\alpha \left[\nu_{jl}(t)\right]^\beta} & \text{if } k \in N^i_j(t) \\ 0 & \text{if } k \notin N^i_j(t) \end{cases} \tag{2}$$

As was previously mentioned, ACO has been applied for symbolic regression and automatic program generation in previous works with promising results. The work in [40] proposed ant colony programming, to solve symbolic regression problems where the construction graph is built from a predefined set of rules. Reference [41] also shows an approach to solve symbolic regression problems, where the construction graph is built over a fully-connected graph of operators and operands. Later, the work [46] suggested using the ACO approach to evolve grammar structures to find classification rules in data mining problems. The Enhanced Generalized Ant Programming (EGAP) was proposed in [47], to solve tree symbolic regression using tree-based grammar representation. In [48], GP is compared with the EGAP approach, concluding that GP statistically improves EGAP in the problems addressed.

In this work, we use ACO to search for SLPs with a balance between accuracy and size. Finding SLPs with reduced size is a topic that has been addressed before in reference [28], which offers an approach to improve accuracy of SLPs for symbolic regression problems in the presence of noisy data, using model regularization. The experimental section of this article compares our approach with the procedure mentioned as a baseline method. *Dynamic Ant Programming* (DAP) [49], another ACO-based approach developed to tackle the bloat problem under the assumption of tree representation of algebraic expressions, will also be included in the experimental section as a baseline method for comparison.

3. Ant Colony Optimization for Straight Line Programs

3.1. Design of the construction graph

As it is mentioned in previous sections, a SLP can be represented as a DAG. The DAG is obtained by means of a simple procedure applied over the SLP grammar rules. Starting from rule U_N, the starting node is created and labelled as U_N, and assigned with the operator of rule U_N. One or two nodes are then created, depending on the arity of the operator, and linked to U_N. If the first (or second, respectively) operand is a terminal symbol, then the node is assigned with the value of the terminal symbol. Otherwise, this procedure is applied recursively over the generated nodes until terminal symbols are reached.

As an example of this procedure, Fig. 1 shows the resulting DAG for the SLP of the Formula (1).

In our approach, since SLPs are grammar-based representations of algebraic expressions, the problem of finding the algebraic expression $\tilde{f}$ that accurately fits output data $\vec{y}$ from a set of input data $\vec{x}$, can be formulated as finding the correct grammar production rules that build a valid SLP. Then, given a maximum number of allowed rules (maximum size N), our approach attempts to find the minimum number of production rules that build a valid SLP (which generates only an algebraic expression), that minimize a loss function $e(\tilde{f}(\vec{x}, \vec{w}), \vec{y})$. Besides, the grammar representation can be translated into

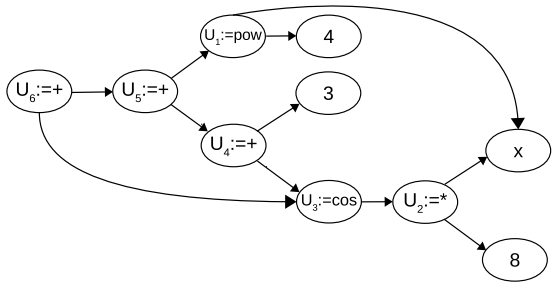


Fig. 1. Example of the corresponding DAG for the SLP example of Figure (1).

a DAG, and this fact suggests that the problem can also be formulated as a graph traverse problem. Thus, the combinatorial problem (S, Ω, e) to be solved in our research work assumes that S is the space of straight line grammar rules with a maximum number of rules equals N, Ω are the constraints of the grammar rules, and e is an error measurement that evaluates the accuracy of an SLP to approximate the desired output data $\vec{y}$.

The construction graph used for ACO algorithms in our research is therefore designed as follows: The starting node is the grammar rule whose antecedent is the non-terminal symbol U_N. The feasible neighborhood of a node U_i is the set of arithmetic operators allowed for building algebraic expressions, $\{O_1, O_2, ..., O_l\}$. Let O_{U_i} be the selected operator for rule U_i. Then, its feasible neighborhood is the set of available operands $\{x_1, x_2, ..., x_n, w_1, w_2, ..., w_k, U_{i-1}, U_{i-2}, ..., U_1\}$, where $x_j, 1 \leq j \leq n$, is the j-th input variable, $w_j, 1 \leq j \leq k$, is the j-th algebraic expression parameter, and $U_j, 1 \leq j < i$, is a non-terminal symbol that makes reference to grammar production rule U_j. Once the first operand has been chosen, then the second operand is selected using the same feasible neighborhood as the one used for the first operand, if O_{U_i} is a binary operator. The feasible neighborhood of the last generated operand is the node corresponding to rule U_{i-1}, and then the process is repeated until the operators and operands of rule U_1 have been generated. This graph representation allows us to store the pheromone trails into 3 matrices:

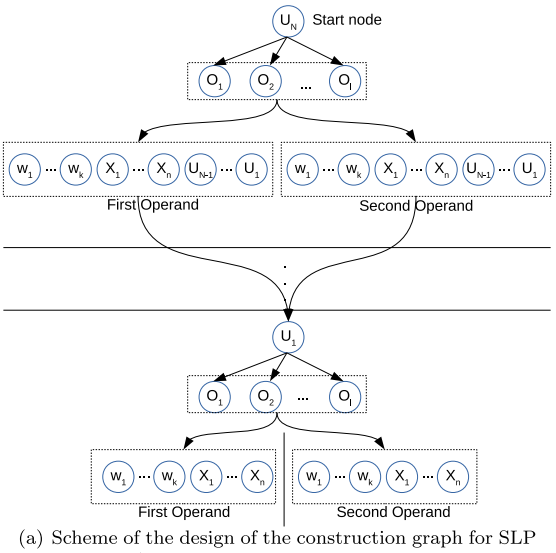
- Matrix $T_o(1..N, 1..l)$, to store pheromone trails from nodes of rules U_i to operator nodes. $T_o(i, j)$ contains the pheromone regarding the selection of operator O_j at rule U_i.
- Matrices $T_{R_1}(1..N, 1..l, 1..n+k+N-1)$, and $T_{R_2}(1..N, 1..l, 1..n+k+N-1)$, to store pheromone trails from operator nodes to the first and second operand nodes, respectively. The value of $T_{R_k}(i, p, j)$ contains the pheromone regarding the selection of symbol j as the value for operand k in the rule U_i, when operator O_p was selected for the rule. Symbol j links to an input variable x_j if $1 \leq j \leq n$, to an algebraic expression parameter w_{j-n} if $n < j \leq n+k$, and to rule U_{j-n-k} if $n+k < j \leq n+k+l$. To ensure verification of the constraints Ω of the combinatorial problem stated, the values $T_{R_k}(i, j) = 0, \forall j : n < j < i+n+k$, i.e. a rule U_i can only contain rule references in the set $\{U_{i-1}, U_{i-2}, ..., U_1\}$.

Fig. 2(a) outlines the general design of the construction graph described. As an example, Fig. 2(b) assumes a set of operators $O = \{O_1, O_2, O_3, O_4, O_5\} = \{+, -, *, /, cos\}$, a set of variables $\vec{x} = \{x\}$, a set of algebraic expression parameters $\vec{w} = \{w_1, w_2\} = \{3, 4\}$, and a maximum set of grammar rules $N = 3$, and describes an example of the graph traverse to obtain the grammar rules $U_3 := U_2 * U_2$, $U_2 := w_1/U_1$, $U_1 := cos(x)$. Firstly, rule U_3 is generated. The operator $*$ is probabilistically selected, and then the first and second operands are generated. In this case, non-terminal symbol U_2 is selected probabilistically for both operands. After rule U_3 is completed, then rule U_2 is generated. The probabilistically selected operator is /, and w_1 and U_1 as the first and second operands, respectively. Finally, rule U_1 is generated, probabilistically choosing the operator cos. Since the operator cos requires a single operand, then only the first operand is generated probabilistically. In this case, terminal symbol x is selected. The construction of the solution is completed, and the algebraic expression encoded into the generated SLP is $(3/cos(x)) * (3/cos(x))$.

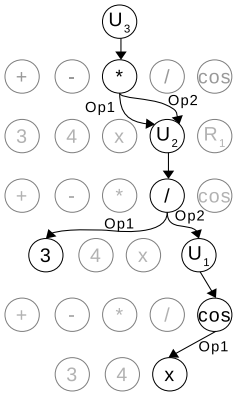
Once the construction graph is designed, the next section describes the components of the algorithm.

3.2. Algorithm design

In this section, we design an ACO-based algorithm to find algebraic expressions $\tilde{f}(\vec{x}, \vec{w})$ encoded as SLPs, with a local search procedure to optimize the algebraic expression parameters $\vec{w}$ simultaneously. The literature offers different methods to solve the problem of parameter $\vec{w}$ estimation, such as [50,51]. Other proposals reduce the problem complexity and select a fixed value for parameters $\vec{w}$ from the beginning [52], although this strategy could shrink the search to less accurate solutions. In our approach, a non-linear least-square method (NLS) [53,54] that minimizes a loss function, is used during the evaluation of each candidate solution $\tilde{f}$ to fit the numerical values for the parameters $\vec{w}$. The loss function used in this work is shown in equation (3), where n stands for the number of data samples, $\vec{x}(i)$ is the i-th input sample, and $y(i)$ is the i-th output sample. Using this fitness measure, the optimal solution $\tilde{f}^*$ has $fitness(\tilde{f}^*) = 1$, while the worst solutions have a fitness value closer to 0. Then, the problem is formulated as a maximization problem.



(a) Scheme of the design of the construction graph for SLP search using ACO



(b) Example of traversing the creation graph.

Fig. 2. Scheme and example of the construction graph for SLP search using ACO.

$$fitness(\tilde{f}) = \frac{1}{1 + \frac{1}{n}\sum_{i=1}^{n}(\tilde{f}(\vec{x}(i), \vec{w}) - y(i))^2} \tag{3}$$

The adaptation of classical ACO implementation [43] to our proposal is shown in Algorithm 1. The procedure starts by initializing the pheromone matrices to an initial value, which is found experimentally by means of a trial-and-error process. Also the best solution, named *Best Ant*, is assigned with an empty value. After initialization, the main algorithm repeats until a stopping criterion is satisfied. In this article, the stopping criterion used in the experimentation is to achieve a number of feasible solutions evaluated.

Each algorithm iteration comprises the following steps: *Solution construction*, *Local search*, *Solution evaluation* and *Pheromone update*:

- The step **Solution construction** builds the path for each ant in the algorithm, following the graph traverse process over the construction graph described in Section 3.1. Unlike classic Ant Colony methods, which use α and β values in equation (2) to define a balance between pheromone trails and heuristic criteria to explore the search space, our model does not rely on heuristic information to build the ant solution. This is because it is difficult to find an appropriate heuristic for operators and operands in symbolic regression, since the fitness of an algebraic expression also depends on the other subexpressions in the solution, and the heuristic of an operator could fail when used in different contexts. For this reason, we set the values $\alpha = 1$ and $\beta = 0$ in our approach. This decision is not uncommon in symbolic regression problems, and it was assumed in previous works as in [49]. The selection of operators and operands is performed probabilistically, using the formula in equation (4), where $N_i(t)$ is the feasible neighborhood of vertex i at algorithm iteration t, $\tau_{ij}(t)$ is the pheromone trail from vertex i to vertex j at iteration t, and $p_{ij}(t)$ is the probability

Algorithm 1 SLP-ACO algorithm.

Require: Input: S_p, the number of ants that run in parallel
Require: Input: $\vec{x} = (x_1, x_2, ..., x_n)$, input data variables for SLP evaluation
Require: Input: $O = \{O_1, O_2, ..., O_l\}$, the operators allowed for the algebraic expression
Require: Input: K, the maximum number of algebraic expression parameters $\{w_1, ...w_k\}$
Require: Input: $\vec{y} = (y_1)$, output data for SLP evaluation
Ensure: Output: $SLP(1...N)$ a sequence of rules that encode the algebraic expression

```
{Initialization}
Set initial operator pheromone T_o(i, j) := T_0, ∀i, j : 1 ≤ i ≤ N, 1 ≤ j ≤ l
Set initial operands pheromone T_R1(i, p, j) = T_0, T_R2(i, p, j) := T_0, ∀i, p, j : 1 ≤ i ≤ N, 1 ≤ p ≤ l, 1 ≤ j ≤ n + k + l
Best Ant := {∅}
t := 1 {Current iteration}
{Main loop}
while No stopping criterion is fulfilled do
  for counter a=1 to S_p do
    {Solution construction}
    Initialize ant_a, the a-th ant
    for i:=N downto 1 do
      Select operator and operands for rule ant_a(i) according to equation (4)
    end for
    {Local search}
    w⃗ := NLS(x⃗, y⃗, ant_a)
    {Solution evaluation}
    Evaluate(ant_a, (w⃗), x⃗, y⃗)
    {Update best solution}
    if fitness(ant_a) > fitness(Best Ant) then
      Best Ant := ant_a
    end if
  end for
  {Pheromone update}
  Perform pheromone update according to equation (5)
  Update next iteration t := t + 1
end while
return Best Ant
```

of moving the ant from vertex i to vertex j in the construction graph. This probability is calculated depending on the type of vertex i on which the ant is located in the construction graph. On one hand, if the ant is located on a vertex corresponding to a non-terminal symbol $U_i \in \{U_1, ...U_N\}$, then $\tau_{ij}(t) = T_o(i, j)$, and the feasible neighborhood of vertex i is the set of operators $N_i(t) = \{O_1, O_2, ..., O_l\}$. On the other hand, if the ant is located at operator p of production rule r, then $\tau_{i,j} = T_{R1}(r, p, j)$ and the feasible neighborhood is $\{x_1, x_2, ..., x_n, w_1, w_2, ..., w_k, U_1, U_2, ..., U_{r-1}\}$. In case the operator p selected for rule r is binary, and the ant is located in vertex i associated with a symbol for the first operand, then $\tau_{i,j} = T_{R2}(r, p, j)$, and the second operand is selected. Otherwise, the remaining possible vertices do not fulfill the constraints in Ω and their selection probability is 0.

$$p_{ij}(t) = \begin{cases} \dfrac{\tau_{ij}(t)}{\sum_{p \in N_i(t)} \tau_{il}(t)} & \text{if } j \in N_i(t) \\ 0 & \text{if } j \notin N_i(t) \end{cases} \tag{4}$$

- The **Local search** performs the algebraic expression parameters $\vec{w}$ optimization, using a non-linear least-square method (NLS) [53,54].
- The **Solution evaluation** process calculates the fitness for each feasible solution found by ants in the current iteration. A solution evaluation is performed as follows: For each i-th input data sample $\vec{x}(i)$, all rules in the solution from U_1 to U_N are evaluated in ascending order, until U_N is reached. Then $\tilde{f}(\vec{x}(i), \vec{w})$ is assigned with the resulting value of the best rule of the SLP. The fitness is calculated using all $\tilde{f}(\vec{x}(i), \vec{w})$ values according to the formula in equation (3).
- The step **Pheromone update** is applied not only to control the amount of pheromone that an ant deposits on the path, but also the pheromone evaporation. Formula (5) shows that the pheromone evaporation rate is controlled using an algorithm parameter $\rho \in [0, 1]$, which reduces the quantity of pheromone proportionally at each iteration.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta fitness_{ij}(Best Ant) \tag{5}$$

In our approach, only the best ant that was found during the search deposits pheromone in proportion to its fitness, as is performed in the Best-Worst ACO approach [55]. The deposition rate is also controlled by an algorithm parameter Δ. The value $fitness_{ij}(Best Ant) = fitness(Best Ant)$ if the edge E_{V_i,V_j} is included in the path over the construction graph obtained by solution $Best Ant$ and E_{V_i,V_j} is not part of a rule classified as *dead code*. The value $fitness_{ij}(Best Ant) = 0$ otherwise. With the term *dead code* we mean all the production rules that are encoded into the SLP solution provided by an ant, but which cannot be derived from U_N. Since these production rules are not used to generate the algebraic expression encoded into the solution, then pheromone deposition is also avoided for these production rules. Formula

(6) shows an example of SLP with size $N = 3$, where the production rule U_2 is *dead code*, since $U_3 \Rightarrow U_1 * U_1 \Rightarrow (x - w_2) * (x - w_2)$, and U_2 cannot be derived from U_3.

$$\begin{aligned} U_1 &\rightarrow x - w_2 \\ U_2 &\rightarrow U_1 + w_1 \\ U_3 &\rightarrow U_1 * U_1 \end{aligned} \tag{6}$$

Finally, with regards to the computational complexity of our proposal, we remark that each solution construction, solution evaluation and pheromone update methods are $O(n)$, where n is the size of the SLP. Then, although the time complexity of the local search procedure is exponential, it is executed under a set of predefined number of iterations, which implies a constant time complexity, as is shown in the experimental section. Consequently, the time complexity of our proposal is $O(n^3 * m)$, where n is the size of the SLP and m is the number of ants. Once the proposed SLP-ACO algorithm has been described, the next section performs an experimental study in a real data scenario.

4. Experiments

Since the final goal of our research is to develop data mining techniques aimed at finding energy consumption models that encompass a balance between accuracy and interpretability, the proposal SLP-ACO is validated using a set of real energy consumption data. In order to prove the potential of our proposal, we used ACO and GA baseline methods to compare the results in terms of not only accuracy but also expression size. Firstly, we have selected an ACO algorithm used to solve symbolic regression problems as baseline method for comparison. More specifically, we used Dynamic Ant Programming (DAP) [49] which uses a tree representation to encode algebraic expressions. For this comparison, we are motivated to study the potential of SLPs over trees and also to verify if the local search method used in SLP-ACO for parameter estimation allows to perform accurate solutions of reduced size. On the other hand, we also compare our proposal with genetic programming algorithms [56]. We have used two genetic programming approaches: the first one uses a local search method for parameter estimation and it is compared with SLP-ACO; the second one does not include a parameter estimator and it is compared with DAP.

In order to clarify the comparison carried out in this section, we named each approach as follows: **DAP** for Dynamic Ant Programming; **SLP-GA** for Genetic Algorithm without using parameter estimation; **SLP-GA-Cte** which uses a local search method for parameter estimation and **SLP-ACO** for our proposal. These algorithms help us to cover a wide variety of proposals that focus on different features regarding our approach -representation, strategies to address the bloat problem, and training models-. Therefore, the main goal of this experimentation attempts to verify the quality of the results provided by each algorithm and study the advantages and limitations of our proposal.

4.1. Application to real scenarios

The real scenario to test our approach is an energy consumption modelling problem that attempts to obtain interpretable and accurate models of energy consumption in public buildings. More specifically, we use a dataset containing the energy consumption of four buildings at the University of Granada, measured hourly in kW/h from March 2013 to October 2015. In order to acquire the energy consumption data, each building is equipped with a Building Automation System (BAS) [57] that retrieves the energy consumption data from sensors and stores the values with their timestamp in a database. The raw energy consumption data series for each building were preprocessed and aggregated to obtain a daily consumption data series, which we use as a starting point in this experimentation. The preprocessing also included filling in missing values due to power cuts, sensor malfunctioning and maintenance tasks, etc. Fig. 3 shows the raw aggregated data series for the four buildings. Finally, to work with uniform data, the data were normalized in the interval [0.0 1.0] (see equation (7), where v_i is the response value, v_{max} is the maximum response observed, v_{min} is the minimum response observed and $r_{normalized}$ is the normalized response). For confidentiality reasons, we are not allowed to provide the data, and the buildings are labelled as B_1, B_2, B_3, B_4, and contain two research centers, a large faculty, and a small faculty.

$$r_{normalized} = \frac{v_i - v_{min}}{v_{max} - v_{min}} \tag{7}$$

The modelling problem that we tackle attempts to explain the relationships on energy consumption data between working days in the same week. Our goal is to provide an interpretable model that can accurately estimate the energy consumption of a working day considering the remaining working days in the same week. The expected outcomes are models of energy consumption which aid understanding of how the energy consumption of different days relates to each other, in order to include these models in other high-level tasks such as anomaly detection and forecasting, for future research. Assuming we name the energy consumption of the working days as d_1, d_2, d_3, d_4, d_5, equation (8) shows that we want to approximate the energy consumption of day i considering the remaining days j_1, j_2, j_3, j_4, where $j_k \neq i \forall k$, and $\vec{w}$ and f are unknown. For this reason, each energy consumption data series initially had 650 values, and was transformed into a multivariate data series with 5 dimensions (one per each working day), with 130 samples (one sample per week).

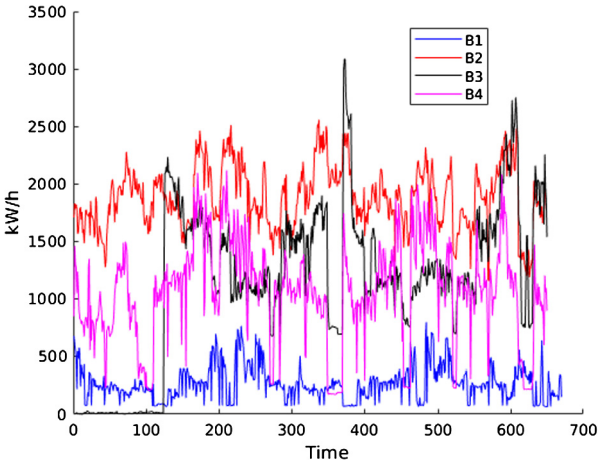


Fig. 3. Energy consumption data series for buildings B_1, B_2, B_3, B_4.

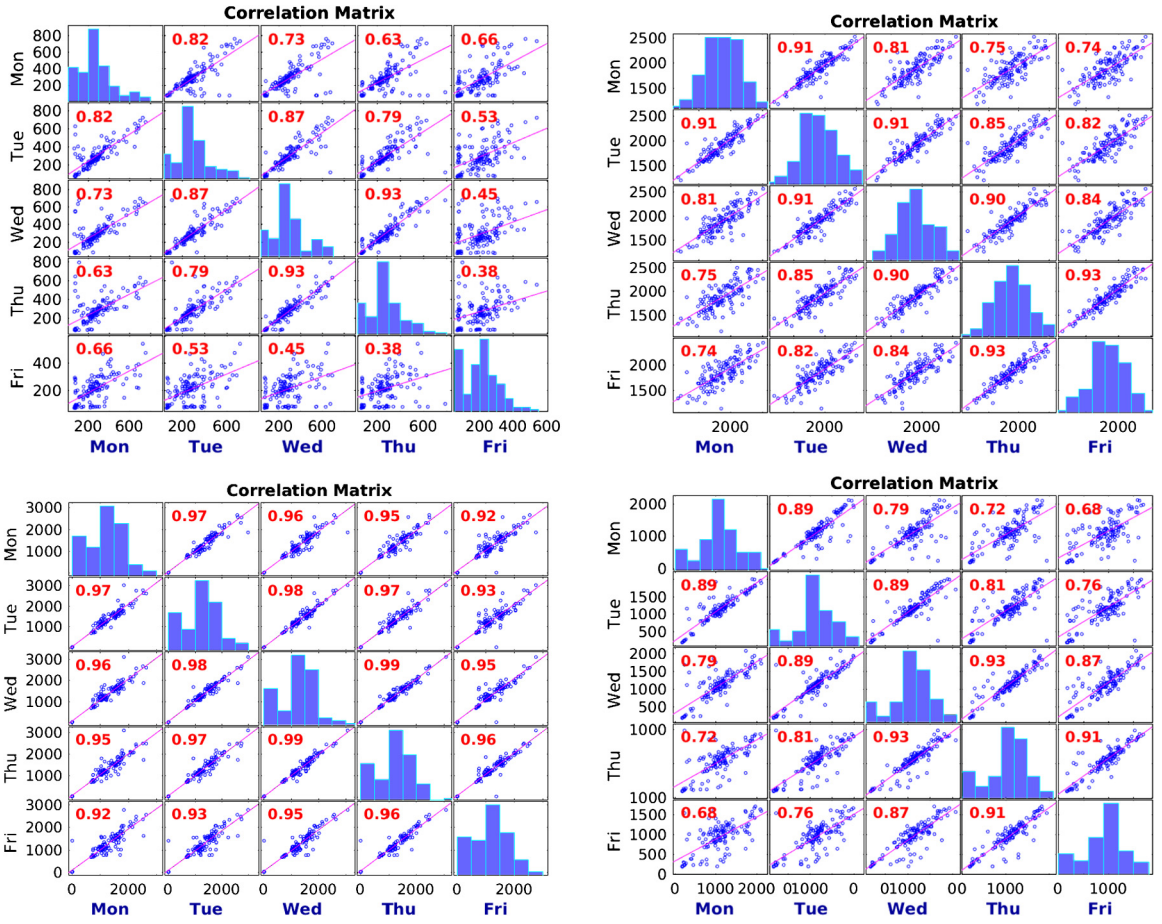


Fig. 4. Correlation matrices of energy consumption for buildings B_1 to B_4, from Monday to Friday. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$d_i = f(d_{j_1}, d_{j_2}, d_{j_3}, d_{j_4}, \vec{w}) \tag{8}$$

There are 20 experiments, to estimate energy consumption of Mondays, Tuesdays, Wednesdays, Thursdays, and Fridays separately, considering the energy consumption of the remaining days in the week as input data, for buildings B_1 to B_4. A preliminary visual and statistical study was first performed, in order to know if there was a correlation between the energy consumption of the working days. Fig. 4 shows the correlation matrices for all buildings and working days. The diagonal

plots of the figures show the histogram of the energy consumption for each working day, and each cell (row i, column j) shows the correlation of day j to day i. Finally, the text in red in the correlation plots shows the correlation coefficient R for the two days being compared. We observe that, as could be expected, there is a high correlation ($R \geq 0.7$) between the energy consumption of two working days in many cases, although there are some cases with an intermediate correlation ($0.3 \leq R < 0.7$). This fact suggests that symbolic regression could be applied to obtain accurate estimation models in the energy consumption modelling problem addressed.

To study generalization capabilities, all datasets were divided into training (first 70% of data) and test (last 30% of data). After that, the training set was used to calculate the fitness value of each method, and the test set was applied over the solution returned from each algorithm execution in order to obtain the results analyzed in this section. For the experimentation, we performed a preliminary extensive experimentation to tune the parameters of both Genetic Algorithms (SLP-GA and SLP-GA-Cte) and Ant Colony approaches (DAP and SLP-ACO). Then, the parameters tuned for both SLP-GA are: 80% of crossover probability and 20% for mutation probability, the population size were established at 70. After that, the experimental configuration for SLP-ACO and DAP are: the minimum value of pheromone rate (ρ_{min}) has been set to 0.01, the evaporation rate (p) were established to 0.5; the number of ants used were 70 and the pheromone value of an inserted node (ρ_{ins}) in DAP is 1. In addition, we allowed a total of 7 parameters ($w_1, w_2, ..., w_7$) for each approach. Moreover, whereas both SLP-GA and DAP have a set of predefined values for each parameter ($w_1 = 1, w_2 = 2, ..., w_7 = 7$), a local search method is used to estimate parameter values for each SLP-GA-Cte and SLP-ACO. Besides, the mathematical operators allowed for all approaches are $\{+, -, *, /, exp, sin, cos, pow, min, max, tan, tanh\}$; the maximum SLP/Tree size are 32 and the stopping criteria are 10000 evaluations. Finally, we performed 30 executions of each algorithms so that we could analyze the results statistically.

Table 1 gathers the results obtained for each algorithm over the test data. Column 1 shows the target working day whose energy consumption is estimated. Then, Columns from 2 to 21 describe the median, best and worst fitness, the average execution time in seconds, and the average size of the solutions provided by DAP, SLP-GA, SLP-GA-Cte and SLP-ACO, respectively. Fitness value is calculated as is described in equation (3) and the size of an algebraic expression is calculated as the number of operators it contains, i.e. the number of non-leaf nodes in tree representation and the number of valid rules in SLP. Moreover, in order to compare the results of each algorithm in terms of fitness and algebraic expression size, we used a statistical test. Due to the results performed by each algorithm does not come from a normal distribution, we decided to use a non parametric test. Consequently, Columns 22 to 25 plots the results of the Kruskal-Wallis (KW) statistical test with a 95% confidence level, to compare each method in terms of fitness values, and Columns 26 to 29 show the solutions regarding the algebraic expression size. The KW test was applied as follows: For each experimentation, the algorithms were sorted from best median fitness/size to worst median fitness/size. A paired KW was applied over the two first algorithms. If significant differences were found ($p\text{-}value < 0.05$), then the algorithm with the best fitness/size was marked with tag 1, and the other one with tag 2, and then the comparison continues with the next algorithm with the best fitness/size. Otherwise, both algorithms were tagged with 1, and the comparison is performed between the algorithm with best median fitness/size and the third algorithm with best median fitness/size. This procedure is applied for all the remaining algorithms results for each problem, until all algorithms have been compared. Finally, for a better analysis of the results in Table 1, we have included the boxplots of the error distribution of all experiments in Fig. 6. Each picture contains the boxplots of the error measure for the algorithms being compared: DAP, SLP-GA, SLP-GA-Cte and SLP-ACO, for the same building and working day.

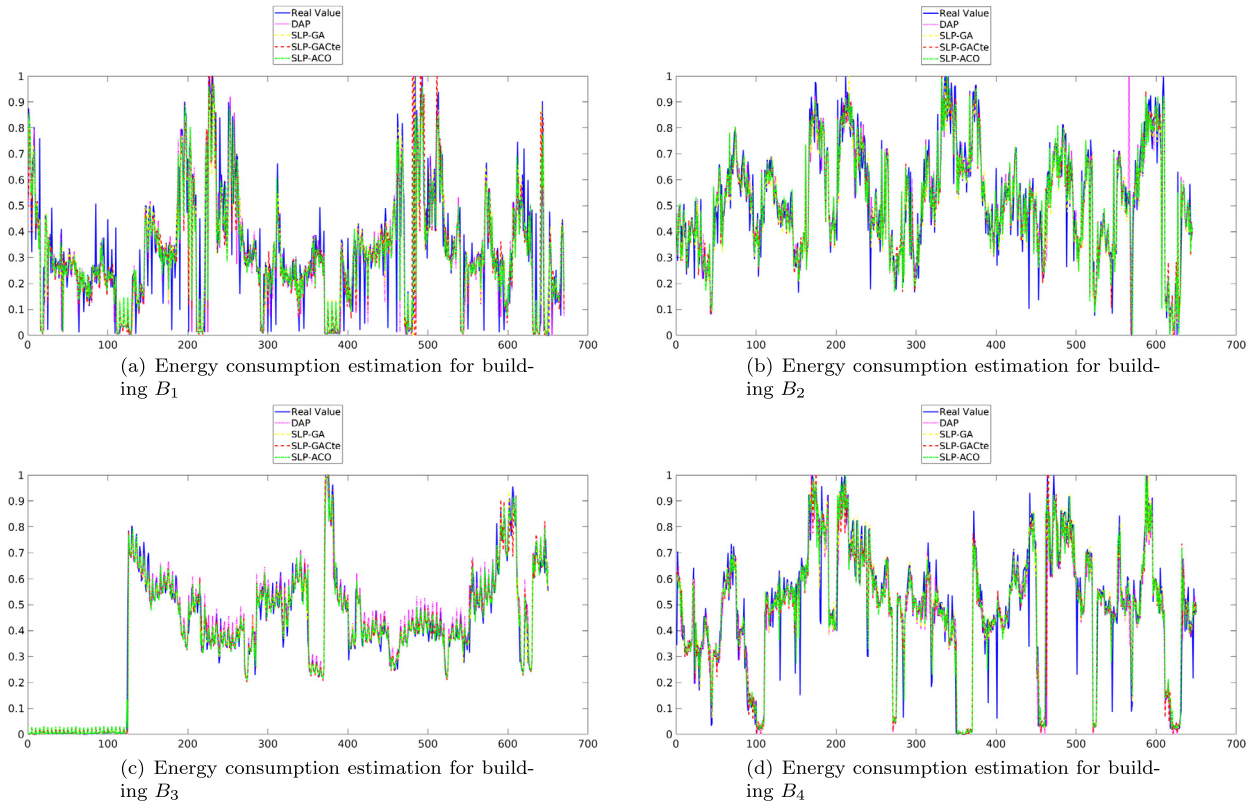
In order to compare baseline methods, the analysis starts by comparing DAP and SLP-GA. Thus, we may observe in Table 1 that SLP-GA achieved better solutions in terms of median values in all cases, whereas DAP performed the worst solution in all problems. With regards of the best fitness, SLP-GA achieved the best solution in 5 experiments, DAP did it in 2 cases and similar solutions were achieved in the remaining 13 experiments. From this analysis we may conclude that SLP-GA is potentially better than DAP, which is supported by the KW test, where SLP-GA achieved better solutions in all cases (shown in columns 22 and 23). The worst solutions provided by DAP may be consequence of the tree representation used to encode algebraic expression and also the local search procedure used by SLP-GA, which may help to avoid local optima and perform better solutions.

On the other hand, if we compare ACO methods (DAP and SLP-ACO) we may observe that SLP-ACO was able to find better solutions in terms of median fitness in all cases, whereas DAP achieved the worst solution in all experiments. Moreover, with regards to the best fitness, SLP-ACO achieved the best solutions in 6 of 20 problems and DAP did it in 1 experiment. These results help us to conclude that the SLP proposal may improve the search of the best algebraic expression, which is supported by KW test in Columns 22 and 25 of Table 1 where we may observe that SLP-ACO performed better solutions than DAP in all experiments. The analysis continues by comparing SLP-ACO and SLP-GA-Cte. Firstly, regarding median fitness, the reader may observe that SLP-ACO was able to achieve the best solution in 1 problem, whereas both approaches performed similar solutions in the remaining 19 experiments. Regarding the best fitness, SLP-GA-Cte found the best solution in 1 experiment and similar solutions were achieved in the remaining 19 experiments. With regards to the worst fitness, SLP-GA-Cte performed worse solutions in 6 problems and SLP-ACO did it in 3 cases. Finally, regarding the KW test we may conclude that SLP-ACO performed better solutions in 4 experiments, SLP-GA-Cte achieved better results in also 4 problems and significant differences were not found in the remaining 12 problems.

From this first analysis we may conclude that SLP approaches are able to find more accurate solutions than tree approaches. Moreover, if we compare SLP-GA-Cte and SLP-ACO approaches, we cannot conclude which approach is better in terms of fitness. On the other hand, regarding the algebraic expression size, we can conclude that ACO approaches are able

Table 1
Results of DAP, SLP-GA, SLP-GA-Cte and SLP-ACO in energy consumption modelling problems.

Working Day	DAP					SLP-GA					SLP-GA-Cte					SLP-ACO					Fitness Test				Size Test			
	Median	Best	Worst	Time	Size	Median	Best	Worst	Time	Size	Median	Best	Worst	Time	Size	Median	Best	Worst	Time	Size								
	Building B1																											
Monday	0.42	0.97	0	13.9	5.9	0.98	0.99	0.98	6	3.93	0.99	0.99	0.97	100.14	11.57	0.99	0.99	0.97	174.29	6.07	3	2	1	2	2	1	4	3
Tuesday	0.32	0.99	0	13.78	4.2	0.98	0.98	0.97	6.02	5.66	0.98	0.99	0.98	93.45	8.00	0.98	0.98	0.98	147.73	3.60	3	2	1	1	2	3	4	1
Wednesday	0.49	0.99	0	13.73	5.43	0.99	0.99	0.99	5.89	4.83	0.99	0.99	0.97	96.96	10.97	0.99	0.99	0.99	175.30	4.83	4	2	1	3	2	1	3	1
Thursday	0.49	0.98	$1*10^{-3}$	13.309	5.7	0.99	0.99	0.69	5.44	4.36	0.99	0.99	0.94	103.84	10.73	0.99	0.99	0.96	160.68	4.40	2	1	1	1	3	1	4	2
Friday	0.75	0.96	0	13.33	7.36	0.97	0.98	0.97	5.48	5.43	0.98	0.98	0.97	103.04	11.97	0.98	0.98	0.94	176.02	6.30	3	2	1	1	3	1	4	2
	Building B2																											
Monday	0.48	0.99	0	14.65	4.8	0.99	0.99	0.99	5.81	6.36	0.99	0.99	0.97	93.91	10.43	0.99	0.99	0.99	165.08	4.70	2	1	1	1	2	3	4	1
Tuesday	0.39	0.99	0	14.4	3.63	0.99	0.99	0.99	5.65	4.93	0.99	0.99	0.98	89.85	9.53	0.99	0.99	0.99	163.35	5.13	3	2	1	1	1	2	4	3
Wednesday	0.15	0.98	0	14.67	4.6	0.99	0.99	0.99	5.86	5.76	0.99	0.99	0.74	96.60	11.37	0.99	0.99	0.99	164.34	4.20	4	3	2	1	2	3	4	1
Thursday	0.83	0.99	0	14.3	4.03	0.99	0.99	0.99	6.21	4.36	0.99	0.99	0.97	89.41	9.33	0.99	0.99	0.98	157.48	3.57	3	2	1	1	2	3	4	1
Friday	0.9	0.99	0	14.3	5.46	0.99	0.99	0.99	6.19	5.6	0.99	0.99	0.99	91.01	8.20	0.99	0.99	0.98	166.70	5.23	3	2	1	1	2	3	4	1
	Building B3																											
Monday	0.46	0.99	0	14.63	5.06	0.99	0.99	0.99	5.49	5.13	0.99	0.99	0.99	85.93	10.60	0.99	0.99	0.99	179.88	4.73	2	1	1	1	2	3	4	1
Tuesday	0.3	0.99	0	13.76	4.63	0.99	0.99	0.99	5.49	4.93	0.99	0.99	0.99	79.09	8.90	0.99	0.99	0.99	157.04	3.93	4	3	2	1	2	3	4	1
Wednesday	0.73	0.99	0	13.83	4.9	0.99	0.99	0.99	5.48	5.03	0.99	0.99	0.99	76.81	7.67	0.99	0.99	0.99	167.11	5.30	2	1	1	1	1	2	4	3
Thursday	0.45	0.99	0	12.96	3.73	0.99	0.99	0.99	5.51	5.43	0.99	0.99	0.99	80.93	9.53	0.99	0.99	0.99	157.58	5.93	3	2	1	1	1	2	4	3
Friday	0.32	0.99	0	13.69	4.93	0.99	0.99	0.99	5.46	6.03	0.99	0.99	0.99	84.68	10.53	0.99	0.99	0.99	173.40	6.37	3	2	1	1	1	2	4	3
	Building B4																											
Monday	0.44	0.98	0	13.17	5	0.98	0.98	0.98	5.47	6.06	0.99	0.99	0.99	93.79	10.90	0.99	0.99	0.99	169.79	4.90	4	3	1	2	2	3	4	1
Tuesday	0.58	0.99	0	13.45	3.96	0.98	0.98	0.98	5.47	4.83	0.98	0.99	0.98	76.45	8.47	0.99	0.99	0.97	175.12	4.47	4	3	2	1	1	3	4	2
Wednesday	0.55	0.99	0	14.37	4.16	0.99	0.99	0.98	5.49	4.76	0.99	0.99	0.99	78.35	9.70	0.99	0.99	0.99	156.33	3.73	3	2	1	2	2	3	4	1
Thursday	0.67	0.99	0	14.37	4.9	0.99	0.99	0.99	5.44	4.1	0.99	0.99	0.99	81.22	9.43	0.99	0.99	0.99	155.00	4.20	3	2	1	1	3	1	4	2
Friday	0.45	0.98	0	14.49	2.9	0.99	0.99	0.99	5.49	4.76	0.99	0.99	0.99	88.52	9.27	0.99	0.99	0.99	168.21	5.17	4	3	2	1	1	2	4	3



(a) Energy consumption estimation for building B_1

(b) Energy consumption estimation for building B_2

(c) Energy consumption estimation for building B_3

(d) Energy consumption estimation for building B_4

Fig. 5. Plots of real data (blue), DAP estimated data (magenta), SLP-GA estimated data (yellow), SLP-GA-Cte estimated data (red) and SLP-ACO estimated data (green) for buildings B_1 to B_4.

to find shorter algebraic expressions with high accuracy. To give support to this conclusion, we may observe the results of the Kruskal-Wallis test in columns 26-27 and 28-29 of Table 1. Firstly, regarding the algebraic expression size of DAP and SLP-GA (columns 26 and 27, respectively), we conclude that DAP performed shorter algebraic expressions in 15 of 20 experiments, whereas SLP-GA achieved shorter algebraic expressions in 5 problems. After that, comparing the results of the algebraic expressions found by SLP-GA-Cte and SLP-ACO (columns 28 and 29, respectively), we may confirm that SLP-ACO achieved shorter solutions in all cases. Nevertheless, if we compare SLP-ACO with SLP-GA we may verify that SLP-GA achieved shorter solutions in 9 cases. In contrast, regarding fitness accuracy, the KW test concludes that SLP-ACO was able to perform better solutions in 13 problems. In this way, we want to highlight the main goal of this research, which attempts to find a balance between accuracy and interpretability. Therefore, we may conclude that SLP-ACO was able to find shorter algebraic expressions with potential accuracy.

With regards to the execution time, we may conclude that ACO approaches need more computational time to find a solution. This fact may be verified in the execution time between DAP vs SLP-GA and SLP-ACO vs SLP-GA-Cte, where both ACO methods need by means two times more than GA approaches to perform a solution. Besides, we want to remark that the local search used in SLP-GA-Cte and SLP-ACO introduces a time overhead of almost 200% regarding the execution time of both methods.

Finally, equations (9) to (12) show an example of the most accurate algebraic expressions found by DAP, SLP-GA, SLP-GA-Cte and SLP-ACO, respectively, to approximate Thursday's energy consumption of building B_4. In these equations, we use the notation shown in equation (8), where d_1, d_2, d_3, d_4, d_5 stand for the energy consumption of Mondays, Tuesdays, Wednesdays, Thursdays, and Fridays, respectively. As we observe, DAP and SLP-GA return simpler algebraic expressions, following by SLP-ACO and SLP-GA-Cte. Nevertheless, although all approaches seem to perform similar fitness, the statistical tests show that SLP-ACO and SLP-GA-Cte were able to achieve better results, but SLP-ACO reached a simpler algebraic expression. As an example of the equation provided by SLP-ACO, an expert could conclude that Thursday's energy consumption can be explained as the combination of the energy consumption of Monday's, Wednesday's and Friday's. The interpretability of this type of algebraic expression could therefore contribute to a better data analysis in higher level decision-making processes. Regarding the accuracy of all methods, Fig. 5 shows that the approximation of the whole data series with the algebraic expressions provided by each method fits the real data correctly, and this fact suggests that SLP-ACO is a promising technique to be used for obtaining a suitable balance between accuracy and solution complexity.

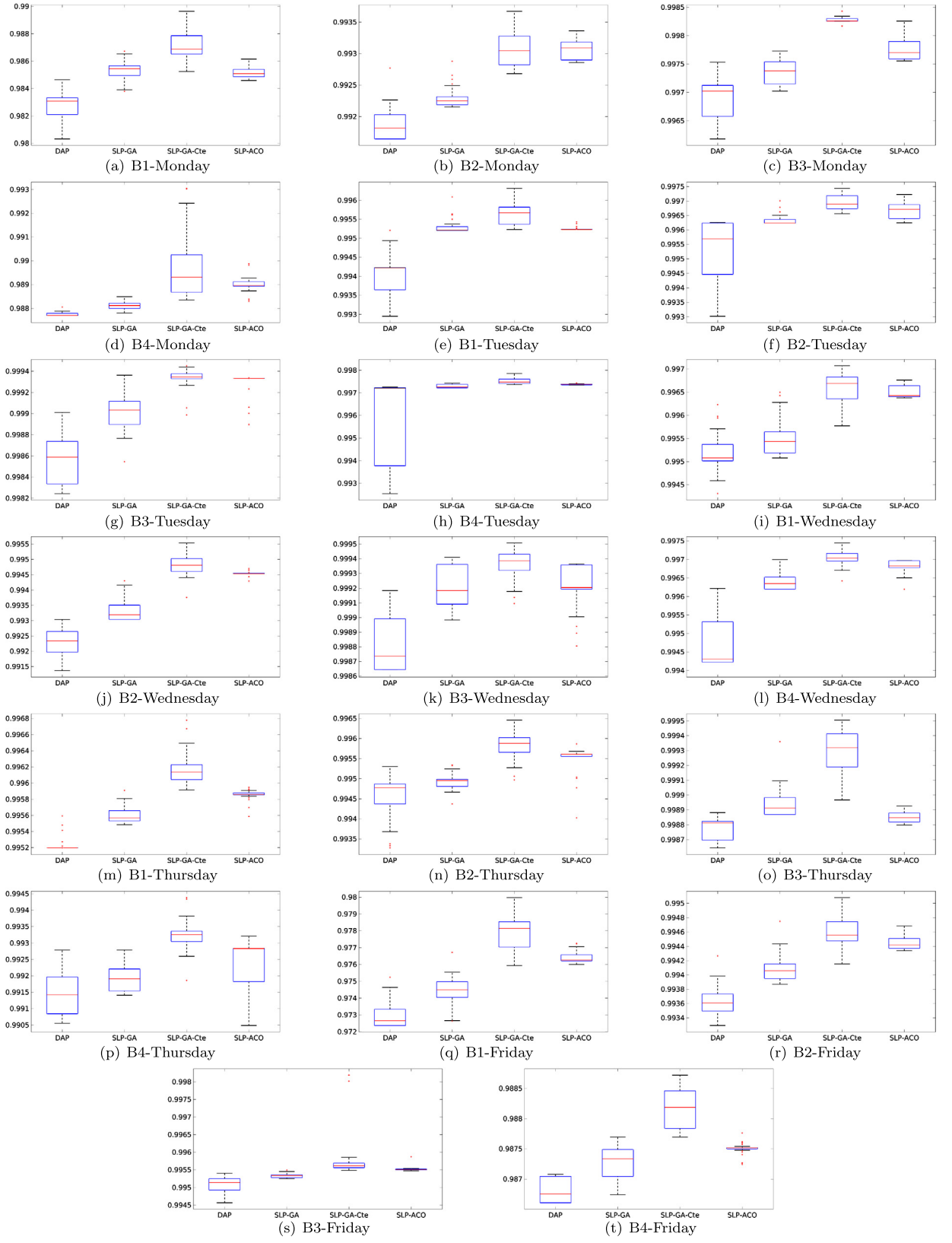


Fig. 6. Boxplots of fitness results for each building, working day and algorithm.

$$d_4^{DAP} = max((log(min(2, d_5)) + 1), d_3) \tag{9}$$

$$d_4^{SLP-GA} = \frac{d_3/1}{exp(d_3 - d_5)} \tag{10}$$

$$d_4^{SLP-GA-Cte} = 0.63 * min(((d_3 + min((((d_3 + (d_5 + (d_3 * -0.95)))^{1.07}) * 0.6), (1.44^{d_5})))^{1.07}), (1.44^{d_5})) \tag{11}$$

$$d_4^{SLP-ACO} = (\frac{1.03^{d_1}}{exp(1.38)} + \frac{1.03^{d_1}}{exp(1.38)}) * ((d_5 + d_3) - log(1.01)) \tag{12}$$

From the aforementioned analysis, we conclude that both SLP-GA-Cte and SLP-ACO provided promising results regarding accuracy in the real symbolic regression problems addressed. On the other hand, SLP-ACO was able to provide solutions with a lower size in all cases, at a cost of increasing the computational time substantially. These lower-size solutions could be more interpretable by an expert, and therefore more suitable for use in higher-level decision making processes than SLP-GA's solutions.

5. Conclusions

In this paper, we have introduced a new algorithm based on Ant Colony Optimization for symbolic regression using Straight Line Programs (SLPs). The approach has been compared with state-of-the-art algorithms with different algebraic representation schemes, and also targeted at minimizing the size of the resulting solutions. The approach has been tested in real energy consumption data. Regarding accuracy, SLP-based algorithms obtained promising results in the problems studied. The linear representation of SLPs allows us to perform a better search over the solution space of algebraic expressions, and also time complexity is reduced when SLPs are trained with genetic programming, compared to tree-based representation schemes. We have also included a local search to fit the resulting algebraic expression parameters inside GA and ACO algorithms. Our experiments show that time complexity is substantially increased using this strategy, but also that accuracy of the resulting solutions can be improved. Regarding the size of the resulting algebraic expressions, ACO based methods provided smaller algebraic expressions than GA approaches. More specifically, DAP was able to find smaller solutions in 15 of 20 problems compared to SLP-GA and SLP-ACO achieved shorter algebraic expression in all experiments, compared to SLP-GA-Cte.

As a general conclusion, the SLP-ACO method proposed in this article has helped to maintain a balance between accuracy and complexity of the solutions provided, and has been tested successfully in real scenarios. Simpler and accurate solutions were obtained using this method, which can help to facilitate a better expert analysis in higher-level decision making processes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the project TIN201564776-C3-1-R.

References

[1] J. Pan, R. Jain, S. Paul, A survey of energy efficiency in buildings and microgrids using networking technologies, IEEE Commun. Surv. Tutor. 16 (2014) 1709–1731.
[2] Z. Wang, R.S. Srinivasan, A review of artificial intelligence based building energy use prediction: contrasting the capabilities of single and ensemble prediction models, Renew. Sustain. Energy Rev. 75 (2017) 796–808.
[3] N. Fumo, M.R. Biswas, Regression analysis for prediction of residential energy consumption, Renew. Sustain. Energy Rev. 47 (2015) 332–343.
[4] J.-S. Chou, A.S. Telaga, Real-time detection of anomalous power consumption, Renew. Sustain. Energy Rev. 33 (2014) 400–411.
[5] W. Cui, H. Wang, A new anomaly detection system for school electricity consumption data, Information 8 (2017) 151.
[6] A. Capozzoli, M.S. Piscitelli, S. Brandi, Mining typical load profiles in buildings to support energy management in the smart city context, Energy Proc. 134 (2017) 865–874.
[7] V. Figueiredo, F. Rodrigues, Z. Vale, J.B. Gouveia, An electric energy consumer characterization framework based on data mining techniques, IEEE Trans. Power Syst. 20 (2005) 596–602.
[8] J. Guan, N. Nord, S. Chen, Energy planning of university campus building complex: energy usage and coincidental analysis of individual buildings with a case study, Energy Build. 124 (2016) 99–111.
[9] T. Chang, W. Fang, L.-F. Wen, Energy consumption, employment, output, and temporal causality: evidence from Taiwan based on cointegration and error-correction modelling techniques, Appl. Econ. 33 (2001) 1045–1056.
[10] Y. Yu, Z. Zou, S. Wang, Statistical regression modeling for energy consumption in wastewater treatment, J. Environ. Sci. 75 (2019) 201–208.
[11] K. Karatzas, N. Katsifarakis, Modelling of household electricity consumption with the aid of computational intelligence methods, Adv. Build. Energy Res. 12 (1) (2018) 84–96.

[12] K. Amber, R. Ahmad, M. Aslam, A. Kousar, M. Usman, M. Khan, Intelligent techniques for forecasting electricity consumption of buildings, Energy 157 (2018) 886–893.
[13] C. Robinson, B. Dilkina, J. Hubbs, W. Zhang, S. Guhathakurta, M.A. Brown, R.M. Pendyala, Machine learning approaches for estimating commercial building energy consumption, Appl. Energy 208 (2017) 889–904.
[14] A.L. Zorita, M.A. Fernández-Temprano, L.-A. García-Escudero, O. Duque-Perez, A statistical modeling approach to detect anomalies in energetic efficiency of buildings, Energy Build. 110 (2016) 377–386.
[15] D.B. Araya, K. Grolinger, H.F. ElYamany, M.A. Capretz, G. Bitsuamlak, An ensemble learning framework for anomaly detection in building energy consumption, Energy Build. 144 (2017) 191–206.
[16] C. Fan, F. Xiao, S. Wang, Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques, Appl. Energy 127 (2014) 1–10.
[17] M.A. Al-Gunaid, M.V. Shcherbakov, D.A. Skorobogatchenko, A.G. Kravets, V.A. Kamaev, Forecasting energy consumption with the data reliability estimatimation in the management of hybrid energy system using fuzzy decision trees, in: 2016 7th International Conference on Information, Intelligence, Systems Applications (IISA), 2016, pp. 1–8.
[18] P. Rodriguez-Mier, M. Mucientes, A. Bugarín, Feature selection and evolutionary rule learning for big data in smart building energy management, Cogn. Comput. 11 (2019) 418–433.
[19] I. Bratko, Machine learning: between accuracy and interpretability, in: Learning, Networks and Statistics, Vienna, 1997, pp. 163–177.
[20] L. Billard, E. Diday, Symbolic regression analysis, in: Classification, Clustering, and Data Analysis: Recent Advances and Applications, Berlin, Heidelberg, 2002, pp. 281–288.
[21] R. Behera, B.B. Pati, B.P. Panigrahi, S. Misra, An application of genetic programming for power system planning and operation, in: International Journal on Control System and Instrumentation, March 2012, pp. 15–20.
[22] M. Bhattacharya, A. Abraham, B. Nath, A linear genetic programming approach for modelling electricity demand prediction in Victoria, in: Hybrid Information Systems, Physica-Verlag HD, Heidelberg, 2002, pp. 379–393.
[23] R. Rueda, M. Cuéllar, M. Delgado, M. Pegalajar, Preliminary evaluation of symbolic regression methods for energy consumption modelling, in: Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, 2017, pp. 39–49.
[24] P.A. Whigham, G. Dick, Implicitly controlling bloat in genetic programming, IEEE Trans. Evol. Comput. 14 (2010) 173–190.
[25] S. Silva, S. Dignum, L. Vanneschi, Operator equalisation for bloat free genetic programming and a survey of bloat control methods, Genet. Program. Evol. Mach. 13 (2012) 197–238.
[26] R. Rueda Delgado, L.G.B. Ruiz, P. Jimeno-Sáez, M.P. Cuellar, D. Pulido-Velazquez, M. Del Carmen Pegalajar, Experimental evaluation of straight line programs for hydrological modelling with exogenous variables, in: Hybrid Artificial Intelligent Systems, Springer International Publishing, 2017, pp. 447–458.
[27] M. Dorigo, M. Birattari, T. Stützle, Ant colony optimization: artificial ants as a computational intelligence technique, IEEE Comput. Intell. Mag. 1 (2006) 28–39.
[28] C.L. Alonso, J.L. Montaña, C.E. Borges, M. de la Cruz Echeandía, A.O. de la Puente, Model regularization in coevolutionary architectures evolving straight line code, in: Computational Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 49–65.
[29] Q. Lu, J. Ren, Z. Wang, Using genetic programming with prior formula knowledge to solve symbolic regression problem, Comput. Intell. Neurosci. 2016 (2016) 1–17.
[30] C. Chen, C. Luo, Z. Jiang, Block building programming for symbolic regression, Neurocomputing 275 (2018) 1973–1980.
[31] B. McKay, M.J. Willis, G.W. Barton, Using a tree structured genetic algorithm to perform symbolic regression, in: First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995, pp. 487–492.
[32] R. Poli, W.B. Langdon, N.F. McPhee, A Field Guide to Genetic Programming, 2008.
[33] 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 (2010) 365–396.
[34] L. Huo, J. Yin, L. Guo, J. Hu, X. Fan, Short-term load forecasting based on improved gene expression programming, in: 2008 4th IEEE International Conference on Circuits and Systems for Communications, 2008, pp. 745–749.
[35] M.F. Brameier, W. Banzhaf, Linear Genetic Programming, 1st edition, Springer Publishing Company, Incorporated, 2010.
[36] C.L. Alonso, J.L. Montaña, J. Puente, C.E. Borges, A new linear genetic programming approach based on straight line programs: some theoretical and experimental aspects, Int. J. Artif. Intell. Tools 18 (2009) 757–781.
[37] R. Rueda Delgado, L.G. Baca Ruíz, M. Pegalajar Cuéllar, M. Delgado Calvo-Flores, M.d.C. Pegalajar Jiménez, A comparison between narx neural networks and symbolic regression: an application for energy consumption forecasting, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications, Springer International Publishing, 2018, pp. 16–27.
[38] F. Benz, T. Kötzing, An effective heuristic for the smallest grammar problem, in: Genetic and Evolutionary Computation Conference, Amsterdam, The Netherlands, July 6-10, 2013, pp. 487–494.
[39] M. Schmidt, H. Lipson, Comparison of tree and graph encodings as function of problem complexity, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, 2007, pp. 1674–1679.
[40] M. Boryczka, Z.J. Czech, Solving approximation problems by ant colony programming, in: Genetic and Evolutionary Computation Conference (GECCO), 2002, pp. 39–46.
[41] J. Green, J.L. Whalley, C.G. Johnson, Automatic programming with ant colony optimization, in: Proceedings of the 2004 UK Workshop on Computational Intelligence, Loughborough University, 2004, pp. 70–77.
[42] S. Bleuler, M. Brack, L. Thiele, E. Zitzler, Multiobjective genetic programming: reducing bloat using spea2, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), vol. 1, 2001, pp. 536–543.
[43] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
[44] B.C. Mohan, R. Baskaran, A survey: ant colony optimization based recent research and implementation on several engineering domain, Expert Syst. Appl. 39 (2012) 4618–4627.
[45] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.
[46] J.L. Olmo, J.R. Romero, S. Ventura, A grammar based ant programming algorithm for mining classification rules, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[47] A. Salehi-Abari, T. White, Enhanced generalized ant programming (egap), in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, New York, NY, USA, 2008, pp. 111–118.
[48] A. Salehi-Abari, T. White, The uphill battle of ant programming vs. genetic programming, in: IJCCI 2009 - International Joint Conference on Computational Intelligence, Proceedings, 2009, pp. 171–176.
[49] S. Shirakawa, S. Ogino, T. Nagao, Automatic construction of programs using dynamic ant programming, in: Ant Colony Optimization, IntechOpen, Rijeka, 2011.
[50] M. Maeder, A.D. Zuberbuehler, Nonlinear least-squares fitting of multivariate absorption data, Anal. Chem. 62 (1990) 2220–2224.
[51] K. Murata, K. Kohno, Nonlinear least-squares regression analysis by a simplex method using differential equations containing Michaelis-Menten type rate constants, Biopharm. Drug Dispos. 10 (1989) 25–34.

[52] D.A. Augusto, H.J.C. Barbosa, Symbolic regression via genetic programming, in: 6th Brazilian Symposium on Neural Networks, Rio de Janeio, Brazil, 2000, pp. 173–178.
[53] D.W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, J. Soc. Ind. Appl. Math. 11 (1963) 431–441.
[54] M. Kommenda, M. Affenzeller, G. Kronberger, S.M. Winkler, Nonlinear least squares optimization of constants in symbolic regression, in: Computer Aided Systems Theory, EUROCAST, Berlin, Heidelberg, 2013, pp. 420–427.
[55] O. Cordón, I.F. de Viana, F. Herrera, Analysis of the best-worst ant system and its variants on the qap, in: Ant Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 228–234.
[56] R. Rueda, L.G.B. Ruiz, P. Jimeno, M.P. Cuellar, D. Pulido-Velazquez, M.C. Pegalajar, Experimental evaluation of straight line programs for hydrological modelling with exogenous variables, in: Hybrid Artificial Intelligent Systems: 12th International Conference, HAIS 2017, La Rioja, Spain, June 21-23, 2017, Proceedings, 2017, pp. 447–458.
[57] T.I. Salsbury, A survey of control technologies in the building automation industry, IFAC Proc. Vol. 38 (2005) 90–100.