Ant Colony based model selection for functional-input Gaussian process regression

José Daniel Betancourt, François Bachoc, Thierry Klein, Fabrice Gamboa

▶ **To cite this version:**

José Daniel Betancourt, François Bachoc, Thierry Klein, Fabrice Gamboa. Ant Colony based model selection for functional-input Gaussian process regression. [Research Report] D3.b, Institut de Mathématiques de Toulouse & Institut Universitaire de France, UMR 5219, Université de Toulouse, CNRS, UPS IMT. 2020. hal-02532713v2

HAL Id: hal-02532713
https://hal.science/hal-02532713v2

Submitted on 17 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Risk-Based System For Coastal Flooding Early Warning
RISCOPE.fr

Ant Colony Based Model Selection for Functional-Input Gaussian Process Regression

Technical Report - April 2020

Deliverable: D3.b
Reference: WP3.2

José Betancourt, François Bachoc, Thierry Klein, Fabrice Gamboa

Financed by

Certified by

RISCOPE research project (RISCOPE.fr) is funded by the French Agence Nationale de la Recherche (ANR) for the period 2017-2021 (ANR, project No. 16CE04-0011). This publication reflects the views only of the author's, and the ANR cannot be considered liable for any use that may be made of the information contained therein.

Recommended citation

Betancourt, J., Bachoc, F., Klein, T., and Gamboa, F. (2020). Technical Report: Ant Colony Based Model Selection for Functional-Input Gaussian Process Regression. Ref. D3.b (WP3.2), *RISCOPE project*.

Contents

1 Introduction **4**

2 ACO-based structural optimization **4**
2.1 Biological inspiration 4
2.2 General Ant Colony System 5
2.3 The ACO-Gp algorithm for model selection 6

3 Analytic test cases **11**
3.1 Black-box functions 11
3.2 Data generation and heuristic setup 12
3.3 Results 13

4 Conclusions **15**

5 Aknowledgments **15**

1 Introduction

Gaussian process models (a.k.a. Kriging models) are one of the preferred choices for metamodeling nowadays [1], competing with a few others like polynomials, splines, generalized linear models and neural networks [2, 3]. While sometimes similar in prediction accuracy to the other cited methods, Gaussian processes present among other advantages: (i) the capability to reproduce complex (and a priori unknown) nonlinear input-output relationships, (ii) the ability to interpolate the observations, and (iii) the interpretability of predictions which include an estimation of the uncertainty at each prediction point.

Gaussian process metamodels were originally developed for scalar inputs, but are now also available for functional inputs. The extension to functional inputs gives rise to a number of questions about the proper way to represent them in the metamodel: (i) which functional inputs are worth keeping as predictors, (ii) which dimension reduction method (DR) is ideal to use (e.g., B-splines, PCA, PLS), (iii) which is a suitable projection dimension, and given our choice to work with Gaussian process metamodels, also the question of (iv) which is a convenient distance to measure similarities between functional input points within the kernel function. Some of these characteristics of the model, along with the type of kernel (e.g., Gaussian, Matérn 5/2) are often chosen a priori, either based on the familiarity of the modeler with a given choice or in results of other metamodeling experiences. However, as shown in [4], the configuration of these characteristics, hereon called ***structural parameters*** of the model, has a strong impact on its prediction capability and the ideal setting might not be trivial. In this report, we introduce **ACO-Gp**, an Ant Colony based algorithm for the structural optimization of Gaussian process models with scalar and/or functional inputs. The architecture of ACO-Gp is tailored to the Gaussian process model, however, its principles are easily extensible to other model selection frameworks.

2 ACO-based structural optimization

Ant colony optimization (ACO) encompasses a large variety of optimization metaheuristics derived from the seminal work of Dorigo et al. in the early 90s [5, 6]. Since then, ACO based heurstics have been proved to give remarkable results in a wide range of optimization problems, including DNA sequencing [7], scheduling [8], protein-ligand docking [9], assembly line balancing [10] and packet-switched routing [11]. ACO has been recognized as one of the most successful research lines in the area of swarm intelligence [12, 13], and always seats beside evolutionary algorithms, iterated local search, simulated annealing, and tabu search among the top metaheuristic techniques [14].

2.1 Biological inspiration

ACO algorithms work based on *stigmergy*, a mechanism for indirect inter-agent communication through traces left in the environment. Ants employ this type of communication to find efficient routes during foraging[1]. The way it works is traditionally explained through a picture similar the one displayed in Figure 1. The frame sequence in alphabetical order illustrates the variant of the *double bridge* experiment performed by Goss et al. in the 80s [15]. In the experiment, the nest of a colony of ants was connected to a food source by two paths, one significantly longer than the other (frame A). At first the ants began to explore

[1]Foraging: searching for wild food resources.

the environment by randomly distributing themselves in the two paths (frame B). Along its way, each ant left pheromone trails noticeable by its mates. As expected, the ants that took the shortest path met the food before (frame C). Most part of the ants that started first the way back to the nest, perceived the larger load of pheromones in the shortest path and went through it. The shortest path kept receiving pheromones at a incrementally higher rate than the longer one, gradually reducing the chances of an ant taking this last (frame D). After some time, the whole colony converged to towards the use of the shortest path (frame E).

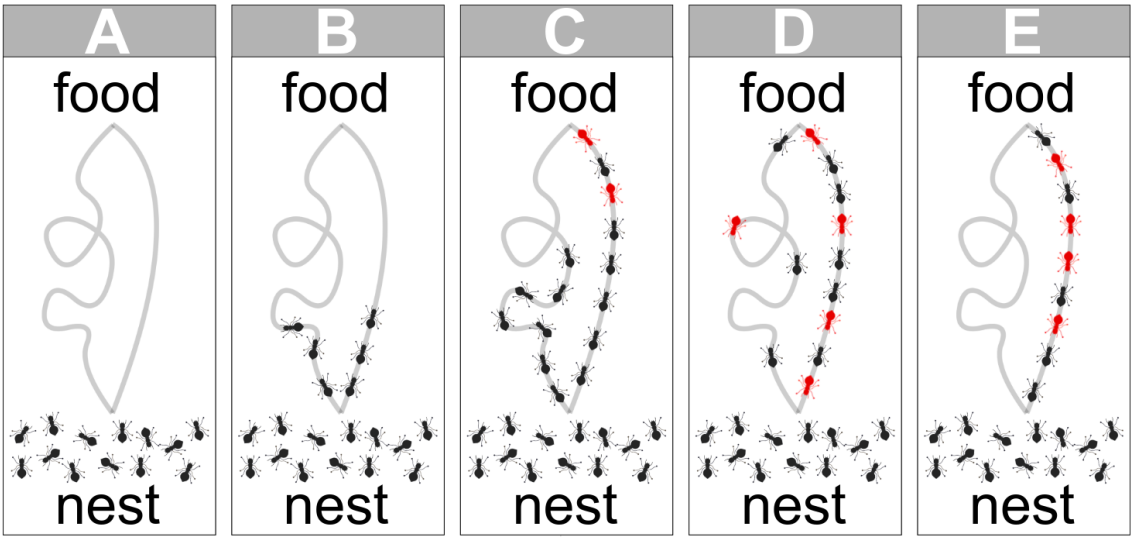


Figure 1: Stigmergy used by ants to find efficient paths towards a food source. Red ants represent the ones going back from the food source to the nest.

2.2 General Ant Colony System

In ACO algorithms, a colony of artificial ants evaluate solutions to the optimization problem at hand. The quality of those solutions is informed to the colony through virtual pheromone trails which help the algorithm to converge towards a high quality solution. To this day there is more than a dozen ACO metaheuristics and probably hundreds of ACO based heuristics[2]. The algorithm proposed here is inspired in the ant colony system (ACS), introduced by Dorigo and Gambardella in the late 90s [16]. In this report we proceed directly with the explanation of our heuristic for model selection. The aforementioned reference is recommended to the reader interested in the original version of ACS.

The ACS operates over a decision network that must be defined in advance based on the structure of the solution space. To start, each ant is located on a base spot and the pheromone value of each link is initialized. Each ant generates a solution to the optimization problem by adding nodes of the decision network to its path according to a pseudo-random system of rules biased by the pheromone loads in the network. Each time an ant traverses a link, a local pheromone update takes place; the pheromone load of the link is slightly reduced in order to foment the diversification of solutions (principle of ***exploration***). Once all ants have made a complete solution, the quality of each solution is evaluated and a global pheromone update occurs; the pheromone load of the links traversed by the best ants is increased in

[2]Metaheuristic vs. heuristic: a metaheuristic is a generic solution technique that can be applied to a broad set of problems; an heuristic is a solution technique designed to resolve a particular problem. An heuristic might be an adaption of an metaheuristic to the particularities of the optimization task at hand.

proportion to the quality of their corresponding solutions, striving for convergence towards high quality solutions (principle of ***exploitation***). The process is then iterated until some stopping conditions are reached. A pseudocode for the ACS is presented in Algorithm 1.

Algorithm 1 Generic ACS structure

```
1: while <stopping conditions remain unsatisfied> do
2:     create a new population of ants
3:     for <i=1:Psize> do
4:         locate ant i at its base spot
5:         tag ant i as partial
6:     end for
7:     while <there are still partial ants> do
8:         randomly pick a partial ant
9:         apply transition rule to select next node in its sequence
10:        reduce pheromone load of chosen link
11:    end while
12:    evaluate the solution made by each ant
13:    increase pheromone load of links in best solutions
14:    P* ← best solution so far
15: end while
16: return P*
```

2.3 The ACO-Gp algorithm for model selection

Decision network

Considering the framework described in Section 1 for a set of ds scalar inputs and df functional inputs our optimization problem consists on making the following decisions:

- State of the i-th scalar input, from {inactive, active};
- State of the j-th functional input, from {inactive, active};
- Projection basis for the j-th functional input, from $\{B_1, \ldots, B_{n_b}\}$;
- Projection dimension for the j-th functional input, from $\{0, \ldots, q_j\}$;
- Distance for the j-th functional input, from $\{D_1, \ldots, D_{n_d}\}$;
- Kernel type, from $\{C_1, \ldots, C_{n_c}\}$,

with $i \in \{1, \ldots ds\}$, $j \in \{1, \ldots df\}$ and q_j the original dimension of input j. The sets $\{B_1, \ldots, B_{n_b}\}$, $\{D_1, \ldots, D_{n_d}\}$ and $\{C_1, \ldots, C_{n_c}\}$ correspond to the basis, distance and kernel families to be considered, in that order. The projection dimension 0 denotes no projection. In order to find a suitable combination of the parameters listed above, we let our artificial ants to move through a network with a structure similar to the one depicted in Figure 2. Such a structure prevents the constitution of senseless solutions (e.g., an input being both, inactive and active) and helps to keep the network data structures considerably simple by only defining strictly necessary links.

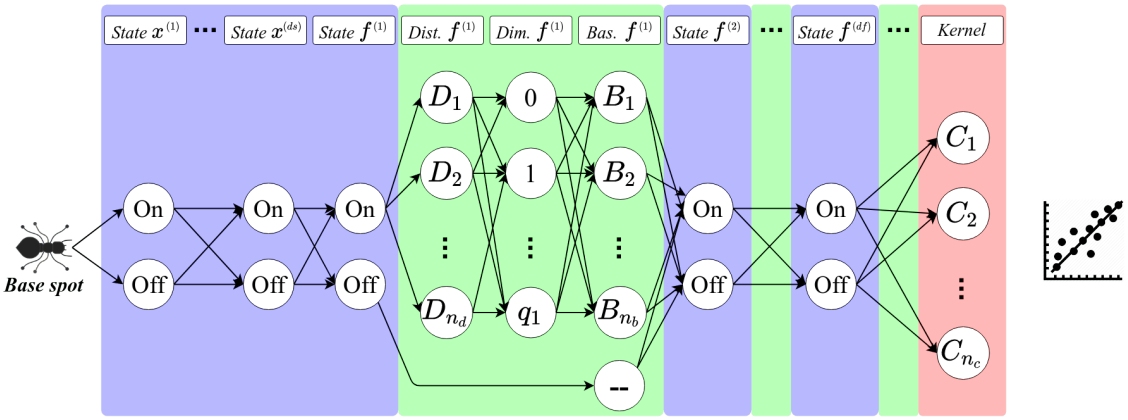


Figure 2: Prototype of the network used in ACO-Gp.

Transition rules

ACO is an iterative algorithm. At each iteration, a group of artificial ants are located at a base spot. Each ant builds a feasible model structure by walking from node to node, always respecting the direction of the links. At each step, an ant selects the next node based on a pseudo-stochastic mechanism defined by (1):

$$\text{rule} = \begin{cases} \text{Rule 1} & \text{if } q \leq q_0, \\ \text{Rule 2} & \text{otherwise,} \end{cases} \tag{1}$$

with q_0 a parameter $\in [0, 1]$ and q a random value from $U(0, 1)$. For ant r located at node a, our transition rules are defined as follows:

Rule 1. Move to the feasible neighbor node with greater pheromone load. Mathematically, it is to move to the node s specified by

$$s = \underset{b \in J_r(a)}{\arg\max}\, \tau_{ab}, \tag{2}$$

with $J_r(a)$ the set of feasible neighbor nodes for ant r located at node a, and τ_{ab} the pheromone load of link (a, b).

Rule 2. Pick the next node based on a probability distribution proportional to the pheromone load of the feasible neighbor nodes. Formally, this can be expressed as moving to node b with probability

$$P(b) = \frac{\tau_{ab}}{\sum_{b \in J_r(a)} \tau_{ab}}, \tag{3}$$

with $J_r(a)$ and τ_{ab} interpreted as in rule 1.

Note that ACO-Gp does not make use of the heuristic visibility value considered in [16] and often present in ACO algorithms. The role of this value, typically denoted η_{ab} for the link (a, b), is to introduce a priori information about the potential benefit of including each link in

the solution. For many optimization problems like routing-oriented and scheduling-oriented ones, the visibility of a link is naturally set to be a function of its inverse generalized cost (see e.g., [17] and [18]). For optimization problems involving categorical variables (such as model selection problem at hand), this set up is often less intuitive since the order of preferences over different levels of the same factor and the relative degree of preference of each level are hard to estimate. For instance, consider the questions of: (i) what would be the order of preference over a set of 5 types of basis families, and (ii) how preferred each basis family should be. Since there is no published evidence of a general superiority of a given basis family irrespective of the regression instance at hand, it could be hard to answer these two questions. In order to include η_{ab} in the algorithm, the same two questions would have to be answered for the distance and the kernel family, and every other feature of the regression model. Hence, we decided to remove the visibility from our model, while keeping in mind that a priori information can also be introduced in the algorithm through the initial pheromone load. This possibility is discussed at the end of Section 2.3.

Pheromone update

As explained in Section 2.2, ACS implements two pheromone update mechanisms – local and global – responsible for the diversification of solutions and the exploitation of acquired knowledge about the structure of high quality solutions. The local pheromone update is triggered each time an ant ads a note to its sequence. The pheromone load of the traverse link is slightly reduced and as a consequence, other ants are less motivated to use the same link in further decisions. In ACO-Gp, the local pheromone update operates on the link (a, b) based on the assignment

$$\tau_{ab} \leftarrow (1 - \rho_l) \cdot \tau_{ab} + \rho_l \cdot \tau_0, \tag{4}$$

where τ_{ab} is the current pheromone load of the link, τ_0 is its initial pheromone load, and $\rho_l \in [0, 1]$ is a parameter that can be interpreted as the pheromone evaporation rate.

On the other hand, global pheromone update takes place each time that a colony becomes complete, i.e., each time all ants in a colony complete a solution. This time, the pheromone load of links belonging to the best ants is increased in proportion to the quality of the corresponding solutions. For each high quality ant, the global pheromone operates on the link (a, b) based on the assignment

$$\tau_{ab} \leftarrow (1 - \rho_g) \cdot \tau_{ab} + \rho_g \cdot \psi, \tag{5}$$

where τ_{ab} is interpreted as in (4), ψ is a measure of the quality of the solution, and $\rho_g \in [0, 1]$ is a parameter that can be interpreted as the learning reinforcement rate. If multiple ants are used in the global update, (4) is applied in an iterated manner over the set of best ants.

Initial pheromone load

The ACS initiates with a base pheromone load on every link. This quantity is modified by the virtual ants during the optimization to communicate actions and learning with their peers. The initial pheromone load must be set with caution, since this value will be determinant on the proper functioning of the algorithm;

- If it is set too low, the heuristic will be prematurely and irreversibly biased towards the best solution of the first iteration, breaking down the learning capability of the system. In addition, the first iteration does not count with any learned information. Thus, the solution at which the system would get stuck might be of regular quality.
- If it is conversely set too high, the system will struggle to converge (if it manages to do so). High quality solutions will not drag the attention they deserve and ants will not be able to focus their exploration around them.

In ACO-Gp, pheromones are implicitly configured to take values exclusively in $[0, 1]$. Thanks to the structure of the updating queries (4) and (5), this is easily achieved by just setting the initial pheromone load of every link in the range $[0, 1]$ and using a quality measure ψ for the solutions in that same range. For the model selection problem, one can pick for instance the Leave-one-out (LOO) cross-validated squared correlation coefficient Q^2_{loocv} (6), or alternatively its hold-out analogous, the predictive squared correlation coefficient Q^2_{hout} (see e.g., [19]).

$$Q^2_{loocv} := \left\lfloor 1 - \frac{\sum_{i=1}^{n.tr}(y_i - \hat{y}_{i,-i})^2}{\sum_{i=1}^{n.tr}(y_i - \bar{y})^2} \right\rfloor_0 , \tag{6}$$

with $(y_i)_{i=1,\dots,n.tr}$ the vector of observed output values, $\bar{y}$ the average of that vector, $\hat{y}_{i,-i}$ the LOO estimation of y_i, and the operator $\lfloor\cdot\rfloor_l$ defined as:

$$\lfloor x \rfloor_l = \begin{cases} x & \text{if } x \geq l, \\ l & \text{otherwise.} \end{cases}$$

Several numerical trials allowed us to identify an initial pheromone load of 0.1, combined with populations of size 10, as a suitable configuration taking into account the setup described in the previous paragraphs. During those trials, we observed the affectation of the pheromone level and the overall behavior of the algorithm along the iterations. In addition, we were able to corroborate the drawbacks associated to excessively low or high τ_0 values, explained in the introductory paragraph of this subsection.

A special treatment for the assignment of the initial pheromone load was given to the links connecting a distance type with a projection dimension (see Figure 2). The framework presented here is general enough to account for any number and type of distance families, however, for practical purposes we adopted the norms $\|\cdot\|_{D,\boldsymbol{\theta_f}}$ and $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$ defined in [4] as a baseline. The selection of one of these two norms not only might have a relevant impact on the predictability of the model, but also on its tractability. The norm $\|\cdot\|_{D,\boldsymbol{\theta_f}}$ requires a single length-scale parameter per functional input, indifferently of its projection dimension. In contrast, for each functional input, the norm $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$ requires as many length-scale coefficients as projection terms. That means that the optimization of the hyperparameters of the model will almost always involve a larger number of decisions variables when using the norm $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$. This norm must be considered as an option as it could be the optimal choice in terms of predictability. However, if the projection dimension is allowed to take too high values, the norm $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$ may imply substantially harder and more time consuming hyperparameters

learning sessions than the norm $\|\cdot\|_{D,\boldsymbol{\theta_f}}$. As a mechanism of regularization, we set up the initial pheromone load for the links pointing out to the norm $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$, based on a loss function of the form

$$\tau_f(x; k_j, \tau_0, \delta_j, w_j) = \begin{cases} \tau_0 \exp\left(-\dfrac{|k_j - 1| - \delta_j}{2\sigma^2}\right) & \text{if } x = 0, \\ \tau_0 & \text{if } 0 < x \leq \delta_j, \\ \tau_0 \exp\left(-\dfrac{|x - 1| - \delta_j}{2\sigma^2}\right) & \text{otherwise,} \end{cases} \tag{7}$$

where k_j is the original dimension of input j, x takes integer values corresponding to its possible projection dimensions, τ_0 denotes the general initial pheromone load of the heuristic, and

$$\sigma^2 = -\frac{w_j^2}{2\log(.5)}.$$

The parameters δ_j and w_j draw the shape of the loss function by specifying the extension of its flat section and the smoothness of its decreasing section (see Figure 3).

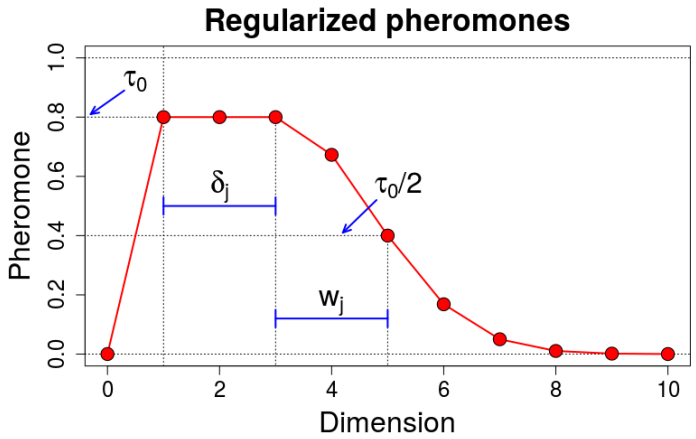
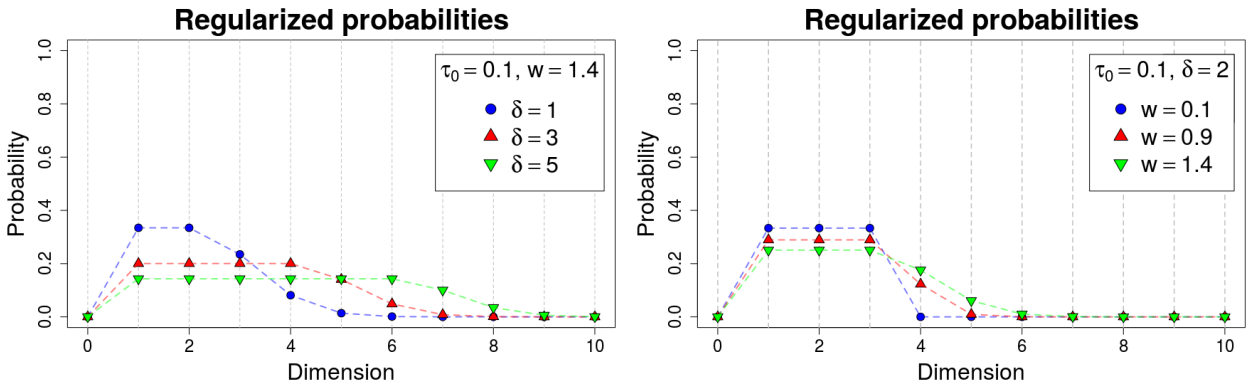


Figure 3: Loss function used for regularization in dimension reduction for functional inputs using the norm $\|\cdot\|_{S,\dot{\boldsymbol{\theta}}_{\boldsymbol{f}}}$. Illustrated for an hypothetical functional input of dimension 10.

The first condition in (7) controls the case where the projection dimension is equal to zero, which as stated earlier in the document denotes no projection. For every functional input, the values of δ_j and w_j can be tuned by observing the normalized loss curve with sum equal to 1. This curve matches the values of the probability pie defined in transition rule 2 (see (3)), and will be used by the ants during the first iteration of the algorithm, each time that this rule is implemented. Some examples of the normalized loss function are provided in Figure 4, where we illustrate the effect of w and δ on the shape of the curve. Suitable w and δ values will vary depending on particularities of the regression task. For instance, if there are many candidate functional inputs, both values could be set relatively low in order to prevent the heuristic from building too heavy models.

As a closing remark, the initial pheromone value can be used to induce preferences on the behavior of the ants. Whenever there is some hint that one level of some feature will perform



(a) Effect of δ on the normalized loss function. (b) Effect of w on the normalized loss function.

Figure 4: Normalized loss function for hypothetical input of dimension 10.

better than the others, this information can be directly placed in the initial pheromone loads of that feature. By doing so, the ants will be stimulated to test more often configurations including the expected best performing level of the feature. The advantage of specifying this information through the pheromones and not through a visibility value (see the discussion on the transition rules) is that in case the induced bias was erroneous, the ants will be able to systematically remove it through the local pheromone update. Conversely, the ants will be able to reinforce the bias through the global pheromone update if they find it fruitful.

3 Analytic test cases

Let us now check the performance of ACO-Gp. We set ourselves in a metamodeling framework where an expensive-to-evaluate computer code is to be substituted by a light-to-run statistical model (see e.g., [20] or [4] for a wider definition of a general metamodeling problem). We consider three analytic black-box functions and we undertake the model selection problem for each of them. We proceed below with the definition of our black-box functions.

3.1 Black-box functions

Let $\mathcal{F}$ be the set of continuous functions from $[0,1]$ to $\mathbb{R}$. Let $\boldsymbol{x} = (x_1, x_2) \in [0,1]^2$ and $\boldsymbol{f} = (f_1, f_2) \in \mathcal{F}^2$ be the vectors of scalar and continuous functional inputs in that order. Then, consider the black-box computer codes specified by the following analytic functions:

- **Analytic black-box 1**

$$\begin{aligned} \mathcal{G}_1 : \quad & [0,1]^2 \times \mathcal{F}^2 \to \mathbb{R}, \\ & (\boldsymbol{x}, \boldsymbol{f}) \mapsto x_1 \sin(x_2) + x_1 \int_0^1 f_1 dt_1 - x_2^2 \left((\max_{T_2} f_2) - (\min_{T_2} f_2) \right). \end{aligned} \tag{8}$$

- **Analytic black-box 2**

$$\begin{aligned} \mathcal{G}_2 : \quad & [0,1]^2 \times \mathcal{F}^2 \to \mathbb{R}, \\ & (\boldsymbol{x}, \boldsymbol{f}) \mapsto x_1 \sin(x_2) + \int_0^1 \exp(x_1 t_1) f_1 dt_1 - x_2^2 \int_0^1 f_2 t_2 dt_1. \end{aligned} \tag{9}$$

- **Analytic black-box 3**

$$\mathcal{G}_3: \quad [0,1]^2 \times \mathcal{F}^2 \to \mathbb{R},$$
$$(\boldsymbol{x}, \boldsymbol{f}) \mapsto x_1 - 2x_2 + 4\int_0^1 t_1 f_1 dt_1 + \int_0^1 f_2 dt_2. \tag{10}$$

Note that the three functions above are totally independent and the only purpose of resolving the model selection problem for all of them is to check the robustness of our proposed algorithm to variations on the structure of the underlying input-output true model.

3.2 Data generation and heuristic setup

Here, we focus on the solution of the model selection problem and we keep the generation of synthetic input data (the experimental design) very simple. For all the three analytic functions we use the same input and output data. We generate the scalar part of the design from a grid over $[0,1]^2$. We assume that the functional inputs f_1 and f_2 are represented by vectors of size 10 and 22, respectively. We made this choice in order to include functional inputs with heterogeneous discretization in the experiment. We sampled all the values of each function randomly from $U(0,1)$. In total, we generated an arbitrary number of 100 input points. For each of them, we computed the corresponding output values using (8), (9) and (10).

As mentioned earlier in the discussion about the initial pheromone load (Section 2.3), a series of numerical trials allowed us to identify the combination $\tau_0 = 0.1$ with a population of size 10 to work properly for our application. During these tests, we also allowed the decision probability boundary q_0, the evaporation rate ρ_l and the learning reinforcement rate ρ_g vary. More specifically, we tried values in the vicinity of $(0.9, 0.1, 0.1)$, the values used for those parameters (in the corresponding order) by Dorigo et al. in [16]. We found the triple $(0.95, 0.1, 0.1)$ to offer a good trade-off between solution discovery and convergence. Finally, we fixed the regularization parameters δ and w at 2 and 1.4, respectively for both functional inputs. This setup produces the normalized loss functions displayed in Figure 5.

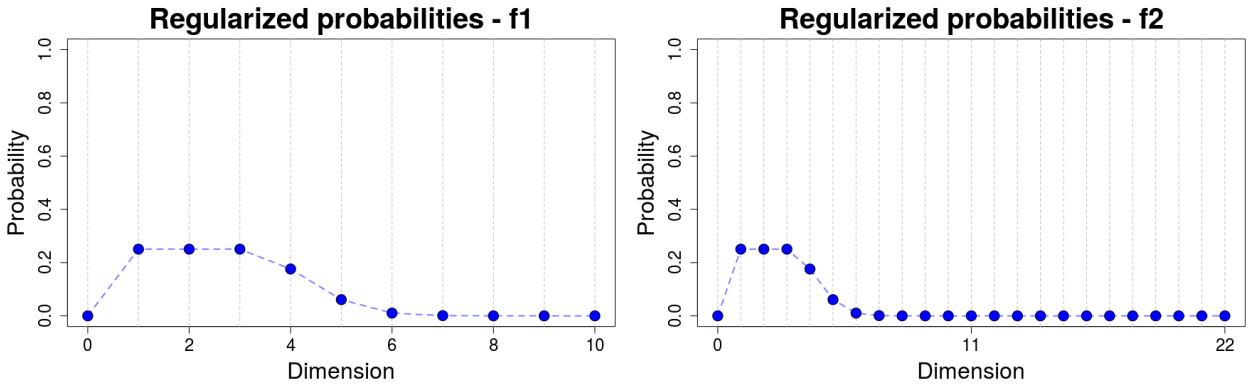


Figure 5: Normalized loss functions used for the analytic cases.

3.3 Results

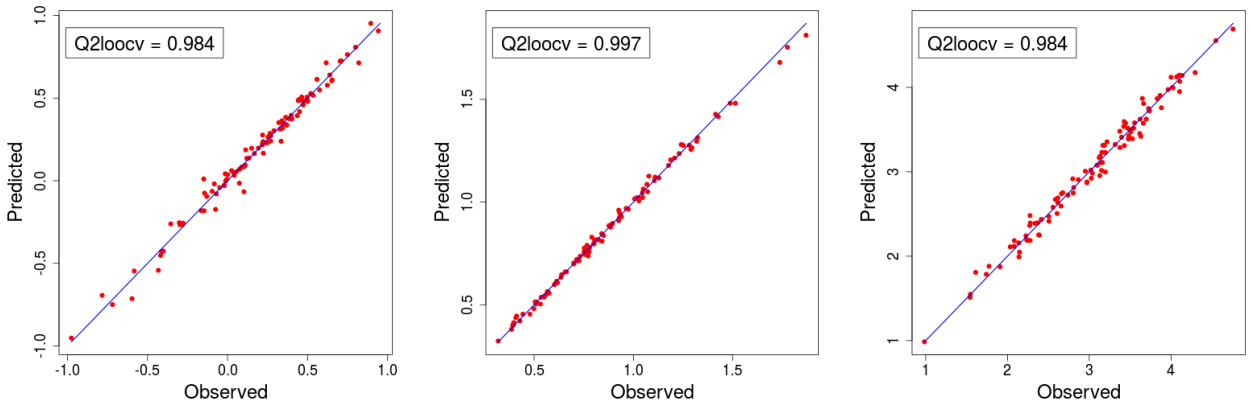
In this section we check the performance of ACO-Gp from three different perspectives:

1) Absolute quality of the selected model;
2) Relative quality of the selected model;
3) Evolution of the solutions found throughout the iterations.

These are better described in the upcoming three subsections. An ideal general number of iterations that will work well for any model selection problem may likely not exist. Since our purpose is merely to show the performance of the heuristic, we used and arbitrary number of 20 iterations. For actual applications where the interest is to find the best possible solution the algorithm can give, we recommend instead using a stopping condition based on processing time. This way, ACO-Gp will search the best possible solution given the time budget.

Absolute model quality

The absolute quality of the model refers to its predictability, regardless of the quality of the other explored models. This dimension of quality can be assessed, for instance, by means of the Q^2_{loocv} or Q^2_{hout} statistics (see the discussion about the initial pheromone load in Section 2.3). Those are conveniently interpretable measures thanks to the fact that they get values in $[0, 1]$ (for any worthwhile model), with a value of 1 indicating perfect fitting and 0 indicating a very poor one. Here we optimize the model structure in terms of the Q^2_{loocv}. Figure 6 displays the calibration plot and the Q^2_{loocv} for the model selected for each of the black-box functions.



(a) Black-box 1. (b) Black-box 2. (c) Black-box 3.

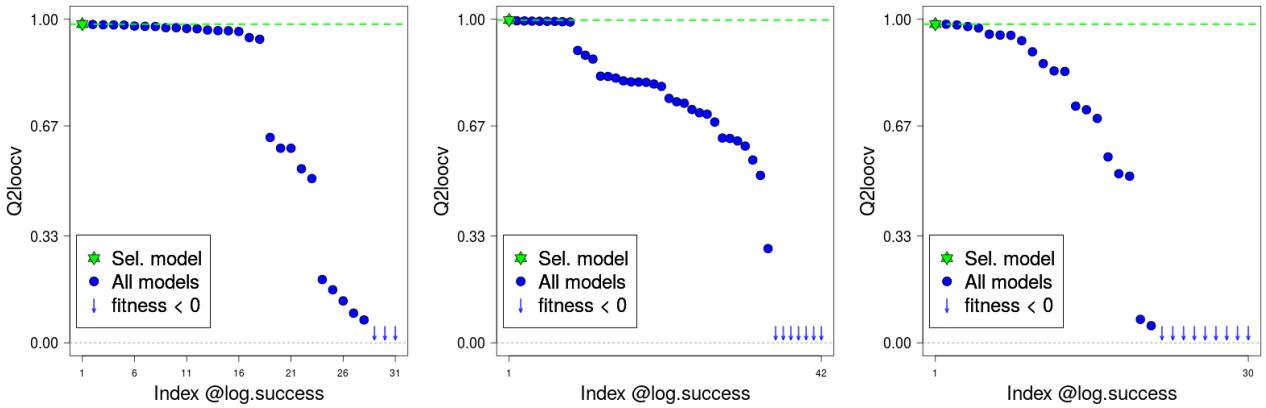
Figure 6: Calibration plot for selected models.

Based on the calibration plots, all the models are good in terms of accuracy and precision. No evident fitting problems (e.g., skewness, heavy tails) are present in any of the models.

Relative model quality

The relative quality of the selected model is assessed by comparing it to the other explored models. This dimension of quality allows to see how special our model is. For instance, a Q^2_{loocv} of 0.95 is not that especial if the vast majority of models had a metrics over 0.92, and similarly, a Q^2_{loocv} of 0.58 is not that bad if most of the models had metrics below 0.35. In

Figure 7 we display the Q^2_{loocv} of all the models explored during the optimization for each of the black-box functions.



(a) Black-box 1. (b) Black-box 2. (c) Black-box 3.

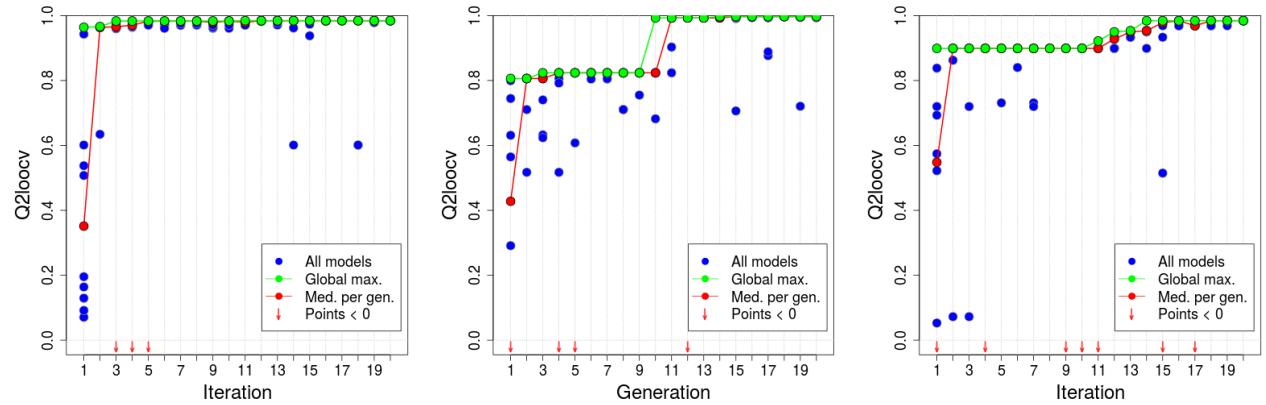
Figure 7: Relative quality of selected models.

In the three cases an important portion of the explored models reported a Q^2_{loocv} that fell well below that of the selected model. Two important takeaways from these plots are:

- The arbitrary selection of the structural parameters is an unsafe move that may be the difference between a poorly performing model and a model of high prediction quality;
- Even if there are multiple relatively good models (as for black-boxes 1 and 2), ACO-Gp will go one step forward and deliver simply the best one.

Evolution

Finally, the analysis of the evolution of the solutions found by ACO-Gp along the iterations helps to verify that its learning mechanism is working properly. In Figure 8 we plot the Q^2_{loocv} of the models explored at each iteration of the heuristic during the optimization for each black-box function, along with the maximum and median Q^2_{loocv} values by iteration.



(a) Black-box 1. (b) Black-box 2. (c) Black-box 3.

Figure 8: Evolution of the heuristic.

In all cases, the maximum and median Q^2_{loocv} progressively improve, and the models gradually converge towards the best solution found. The sporadic drops of the median are just an effect

of the randomness and the degree of exploration requested via the parameter q_0 (see (1)). This phenomenon is by no means a bad thing, since it prevents the algorithm from getting trapped in local optima and allows it to keep improving the best solution as we see in the plots. The optimizations presented here ran in 77.7, 30.4 and 25.8 seconds, for black-boxes 1, 2 and 3, respectively.

4 Conclusions

This technical report introduces the ACO-Gp Ant Colony based algorithm for structural configuration of Gaussian process regression models. After almost 30 decades of their introduction to the scientific community, the ACO algorithms remain powerful optimization tools for concurrent research problems of notable relevance. In this report, we validate the ability of our algorithm to develop high quality regression models through three analytic test cases. In all of them the results were highly satisfactory. Some tests have already been done with ACO-Gp applied to the RISCOPE data (see [4] and [21] for more details on the data) and results seem promising as well. ACO-Gp was implemented in the funGp R package [22] for Gaussian process regression with scalar and functional inputs, which was developed as part of the RISCOPE project. The funGp package is currently downloadable from GitHub and CRAN .

5 Aknowledgments

This study was conducted in the frame of the RISCOPE project, funded by the French Agence Nationale de la Recherche (ANR). We thank the ANR for this support. The application case addressed in RISCOPE motivated the development of the algorithm presented here. We are grateful to Déborah Idier and Jérémy Rohmer from BRGM for their contributions regarding the metamodeling aspects of RISCOPE.

References

[1] A. Marrel, B. Iooss, F. Van Dorpe, and E. Volkova, "An efficient methodology for modeling complex computer codes with gaussian processes," *Computational Statistics & Data Analysis*, vol. 52, no. 10, pp. 4731–4744, 2008.

[2] R. A. Beezer, T. Hastie, R. Tibshirani, and J. F. Springer, "The elements of statistical learning: Data mining, inference and prediction. by," 2002.

[3] K.-T. Fang, R. Li, and A. Sudjianto, *Design and modeling for computer experiments*. Chapman and Hall/CRC, 2005.

[4] J. Betancourt, F. Bachoc, T. Klein, D. Idier, R. Pedreros, and J. Rohmer, "Gaussian process metamodeling of functional-input code for coastal flood hazard assessment," *Reliability Engineering & System Safety*, p. 106870, 2020.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," 1991.

[6] M. Dorigo, "Optimization, learning and natural algorithms," *PhD Thesis, Politecnico di Milano*, 1992.

[7] C. Blum, M. Y. Vallès, and M. J. Blesa, "An ant colony optimization algorithm for dna sequencing by hybridization," *Computers & Operations Research*, vol. 35, no. 11, pp. 3620–3635, 2008.

[8] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 sixth annual ChinaGrid conference*, pp. 3–9, IEEE, 2011.

[9] O. Korb, T. Stützle, and T. E. Exner, "An ant colony optimization approach to flexible protein–ligand docking," *Swarm Intelligence*, vol. 1, no. 2, pp. 115–134, 2007.

[10] A. S. Simaria and P. M. Vilarinho, "2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines," *Computers & Industrial Engineering*, vol. 56, no. 2, pp. 489–506, 2009.

[11] D. Zhao, L. Luo, and K. Zhang, "An improved ant colony optimization for communication network routing problem," in *2009 Fourth International on Conference on Bio-Inspired Computing*, pp. 1–4, IEEE, 2009.

[12] E. Bonabeau, M. Dorigo, D. d. R. D. F. Marco, G. Theraulaz, G. Théraulaz, *et al.*, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, no. 6791, pp. 39–42, 2000.

[14] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[15] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[16] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[17] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, and A. Gray, "The minimum power broadcast problem in wireless networks: an ant colony system approach," in *proceedings of the IEEE Workshop on Wireless Communications and Networking*, 2002.

[18] M. Garmsiri and M. R. Abassi, "Resource leveling scheduling by an ant colony-based model," *Journal of Industrial Engineering International*, vol. 8, no. 1, p. 7, 2012.

[19] J. Nilsson, S. de Jong, and A. K. Smilde, "Multiway calibration in 3d qsar," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 11, no. 6, pp. 511–524, 1997.

[20] A. Marrel, B. Iooss, B. Laurent, and O. Roustant, "Calculations of sobol indices for the gaussian process metamodel," *Reliability Engineering & System Safety*, vol. 94, no. 3, pp. 742–751, 2009.

[21] D. Idier, A. Aurouet, F. Bachoc, A. Baills, J. Betancourt, J. Durand, R. Mouche, J. Rohmer, F. Gamboa, K. T, J. Lambert, G. Le Cozannet, S. Le Roy, J. Louisor, R. Pedreros, and A. Véron, "Toward a user-based, robust and fast running method for

coastal flooding forecast, early warning, and risk prevention," *Journal of coastal research*, vol. 95, pp. 11–15, 2020.

[22] J. Betancourt, F. Bachoc, and T. Klein, *funGp: Gaussian Process Models for Scalar and Functional Inputs*, 2020. R package version 0.1.0.