Pareto-Based Multi-output Model Type Selection

Dirk Gorissen[1], Ivo Couckuyt[1], Karel Crombecq[2], and Tom Dhaene[1]

[1] Ghent University - IBBT, Department of Information Technology (INTEC), Gaston Crommenlaan 8, Bus 201, 9050 Ghent, Belgium
[2] University of Antwerp, Dept. of Computer Science, Middelheimlaan 1, 2020 Antwerp, Belgium

Abstract. In engineering design the use of approximation models (= surrogate models) has become standard practice for design space exploration, sensitivity analysis, visualization and optimization. Popular surrogate model types include neural networks, support vector machines, Kriging models, and splines. An engineering simulation typically involves multiple response variables that must be approximated. With many approximation methods available, the question of which method to use for which response consistently arises among engineers and domain experts. Traditionally, the different responses are modeled separately by independent models, possibly involving a comparison among model types. Instead, this paper proposes a multi-objective approach can benefit the domain expert since it enables automatic model *type* selection for each output on the fly without resorting to multiple runs. In effect the optimal model complexity and model type for each output is determined automatically. In addition a multi-objective approach gives information about output correlation and facilitates the generation of diverse ensembles. The merit of this approach is illustrated with a modeling problem from aerospace.

1 Introduction

Regardless of the rapid advances in High Performance Computing (HPC) and multicore architectures, it is rarely feasible to explore a design space using high fidelity computer simulations. As a result, data based surrogate models (otherwise known as metamodels or response surface models) have become a standard technique to reduce this computational burden and enable routine tasks such as visualization, design space exploration, prototyping, sensitivity analysis, and optimization.

This paper is concerned with efficiently and automatically generating accurate *global* surrogates (valid over the complete design space) using a minimal number of computationally expensive simulations (as opposed to Surrogate Based Optimization (SBO)). Optimization of the simulation output is not the main goal, rather we are concerned with optimization of the surrogate model parameters (hyperparameter optimization). Remark also that since data is expensive to obtain, it is impossible to use traditional, one-shot experimental designs. Data points must be selected iteratively, there where the information gain will be the greatest (active learning). An important consequence hereof is that the task of finding the best approximation (= an optimization problem over the model hyperparameters) becomes a dynamic problem instead of a static one, since the optimal model parameters will change as the amount and distribution of data points changes.

E. Corchado et al. (Eds.): HAIS 2009, LNAI 5572, pp. 442–449, 2009.

In engineering design simulators are typically modeled on a per-output basis [1]. Each output is modeled independently using separate models (though possibly sharing the same data). Instead of this single objective approach, the authors propose to model the system directly using multi-objective algorithms. This benefits the practitioner by giving information about output correlation, facilitating the generation of diverse ensembles (from the Pareto-optimal set), and enabling the automatic selection of the best model type for each output without having to resort to multiple runs. This paper will discuss these issues and apply them to an aerospace modeling problem.

2 Modeling Multiple Outputs

It is not uncommon that a simulation engine has multiple outputs that all need to be modeled (e.g., [2]). Also many Finite Element (FE) packages generate multiple performance values for free. The direct approach is to model each output independently with separate models (possibly sharing the same data). This, however, leaves no room for trade-offs nor gives any information about the correlation between different outputs. Instead of performing multiple modeling runs (doing a separate hyperparameter optimization for each output) both outputs can be modeled directly if models with multiple outputs are used in conjunction with a multi-objective optimization routine. The resulting Pareto front then gives information about the accuracy trade-off between the outputs and allows the practitioner to choose the model most suited to the particular context. In addition, the final Pareto front enables the generation of diverse ensembles, where the ensemble members consist of the (partial) Pareto-optimal set (see references in [3]). This way all the information in the front can be used. Rainfall runoff modeling and model calibration in hydrology [4] are examples where this multi-objective approach is popular. Models are generated for different output flow components and/or derivative measures and these are then combined into a weighted ensemble or fuzzy committee.

In particular a multi-objective approach enables integration with the automatic surrogate model type selection algorithm described in [5]. This enables automatic selection of the best model type (Kriging, neural networks, support vector machines (SVM), ...) for each output automatically, without having to resort to multiple runs. This is the topic of this paper.

3 Related Work

There is a growing body of research available on multi-objective hyperparameter optimization and model selection strategies. An extensive and excellent overview of the work in this area is given by Jin et. al. in [3]. By far the majority of the cited work uses multi-objective techniques to improve the training of learning methods. Typically an accuracy criterion (such as the validation error) is used together with some regularization parameter or model complexity measure in order to produce more parsimonious models [6]. In engineering design this is closely related to the *"The 5 percent problem"* [7], which arises since single criteria are inadequate at objectively gaging the quality of an approximation model [8]. Another topic that has been the subject of extensive research is that of multi-objective surrogate based optimization (MOSBO). Two examples are

ParEGO [9], and the surrogate based variants of NSGA-II [10]. Though the research into MOSBO is still very young, an excellent overview of current research is already available in [11].

When applying surrogate modeling methods, a recurring question from domain experts is *"Which approximation method is best for my data?"*. Or, as [11] states: *"Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice"*. Thus an algorithm to automatically solve this problem is very useful [12]. This is also noticed by [10] who compare different surrogate models for approximating each objective during optimization. The primary concern of a domain expert is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead. Model selection, model parameter optimization, sampling strategy, etc. are of lesser or no interest to them. Little work has been done to tackle this problem. A solution based on Particle Swarm Optimization (PSO) has been proposed for classification in [13] and a GA based solution for regression in [5]. With this paper we extend the GA based solution so the benefits of automatic model type selection can be carried over to the multi-output case.

4 Core Algorithm

The approach of this paper is built around an island model GA and is illustrated in figure 1. The general idea is to evolve different model types cooperatively in a population and let them compete to approximate the data. Models that produce more accurate fits will have a higher chance of propagating to the next generation. The island model is used since it is the most natural way of incorporating multiple model types into the evolutionary process without mixing them too fast. It also naturally allows for hybrid solutions as will be discussed later. Different sub-populations, called *demes*, exist (initialized differently) and sporadic migration can occur between islands allowing for the exchange of genetic material between species and inter-species competition for resources. Selection and recombination are restricted per deme, such that each subpopulation may evolve towards different locally optimal regions of the search space.

The implementation is based on the Matlab GADS toolbox, which itself is based on NSGA-II. Recall from section 1 that data points are selected iteratively (active learning). The general control flow is as follows (see [5] and the Matlab documentation for implementation details): After the initial Design Of Experiments (DOE) has been calculated, an initial sub-population M_i is created for each model type T_i ($i = 1,..,n$ and $M = \bigcup_{i=1}^{n} M_i$). The exact creation algorithm is different for each model type so that model specific knowledge can be exploited. Subsequently, each deme is allowed to evolve according to an elitist GA. Parents are selected using tournament selection and offspring undergo either crossover (with probability p_c) or mutation (with probability $1 - p_c$). The models M_i are implemented as Matlab objects (with full polymorphism) thus each model type can choose its own representation and mutation/crossover implementations. While mutation is straightforward, the crossover operator is more involved since migration causes model types to mix. This raises the question of how to meaningfully recombine two models of different type (e.g., a SVM with a rational function). The solution is to generate a hybrid model by combining both into an ensemble. Once

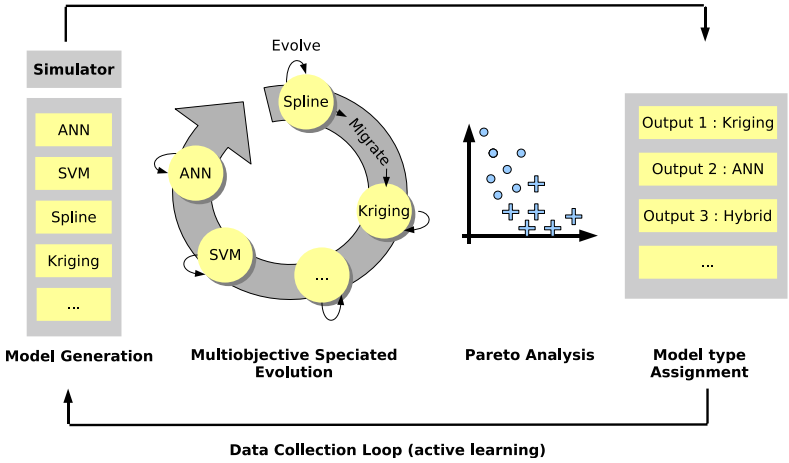

Fig. 1. Automatic multi-objective model generation

every deme has gone through a generation, migration between individuals is allowed to occur at migration interval m_i, with migration fraction m_f and migration direction m_d (a ring topology is used). Finally, an extinction prevention algorithm is used to ensure no model type can disappear completely from the population.

The fitness function calculates the quality of the model fit, according to some generalization estimator ξ (e.g., cross validation). Note that ξ itself can also constitute multiple criteria (see [7] for a discussion on this). Once the GA has terminated, control passes back to the main global surrogate modeling algorithm. At that point M contains the best set of models that can be constructed for the given data. If M contains at least one model that satisfies the user defined targets for ξ the main loop terminates. If not, a new set of maximally informative sample points is selected based on several criteria (quality of the models, non-linearity of the response, etc.) and scheduled for evaluation. Once new simulations become available the GA is resumed.

Note that sample evaluation and model construction/hyperparameter optimization run in parallel to allow an optimal use of computational resources. Some initial results with this algorithm can be found in [7].

5 Langley Glide-Back Booster (LGBB)

For this paper we consider a modeling problem from aerodynamics. NASA's Langley Research Center is developing a small launch vehicle (SLV) [2] that can be used for rapid deployment of small payloads to low earth orbit at significantly lower launch costs, improved reliability and maintainability. In particular, NASA is interested in the aerodynamic characteristics (lift, drag, pitch, side-force, yaw, roll) as a function of three inputs (Mach number, angle of attack, and side slip angle). Simulations are performed with a Cart3D flow solver with a running time of 5-20 hours on a high end workstation. A fixed data set of 780 adaptively chosen points was generated for metamodeling purposes. Thus the active learning loop is switched off.

6 Experimental Setup

The platform of choice for the implementation and experiments described in this paper is the Matlab **SU**rrogate **MO**deling Toolbox (SUMO Toolbox) v6.1 [1]. The toolbox is available for download at `http://www.sumo.intec.ugent.be` to allow for full reproduction of these results. For the modeling the following model types are used: Artificial Neural Networks (ANN), rational functions, Kriging models [14], and RBF LS-SVMs [15]. As stated in subsection 4 the result of a heterogeneous recombination will be an ensemble. So in total 5 model types will be competing to approximate the data. Remember that all model parameters are chosen automatically as part of the GA. No user input is required, the models are generated automatically.

The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization (300 epochs). The topology and initial weights are determined by the Genetic Algorithm (GA). For the LS-SVMs the c and σ are selected by the GA as are the correlation parameters for Kriging (with a linear regression and Gaussian correlation function). The rational functions are based on a custom implementation, the free parameters being the orders of the two polynomials, the weights of each parameter, and which parameters occur in the denominator. A full explanation of the genetic operators used for each model would consume too much space. All the code is available as part of the SUMO Toolbox and details can be found in the associated technical reports.

The population size of each deme type is set to 15 and the GA is run for 50 generations. The migration interval m_i is set to 5, the migration fraction m_f to 0.1 and the migration direction is *forward* (copies of the m_f best individuals from island i replace the worst individuals in island $i+1$). The fitness function is defined as the Bayesian Estimation Error Quotient ($BEEQ(y,\tilde{y}) = \frac{\sum_{i=1}^{n}|y_i-\tilde{y}_i|}{\sum_{i=1}^{n}|y_i-\bar{y}|}$) calculated on a 20% validation set where the minimal distance between validation points is maximized. The variables y,$\tilde{y}$ and $\bar{y}$ represent the true, predicted and mean true values respectively.

7 Results

For ease of visualization all outputs were modeled in pairs. Figure 2 shows the resulting search trace for three representative combinations (the others were omitted to to save space). The algorithm has been proven to be quite robust in in its model selection ([5,7]) thus we can safely infer from these results.

Let us first regard the $lift - drag$ trace. We notice a number of things. First all the models are roughly on the main $x = y$ diagonal. This means that a model that performs well on $lift$ performs equally well on $drag$. This implies a very strong correlation between the behavior of both outputs, which can actually be expected given the physical relationship between aerodynamic lift and drag. Only the rational models do not consistently follow this trend but this has probably more to do with their implementation as will be discussed later. Since all models are on the main diagonal the model type selection problem has become a single objective problem. It turns out that the ANN models (or actually an ensemble of ANN models if you look closely) are the best to

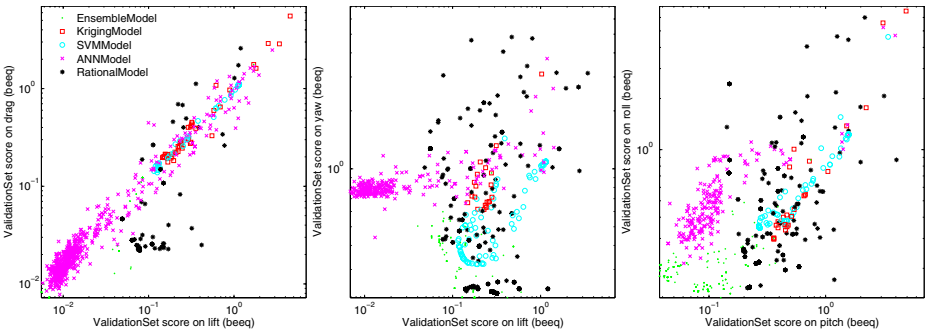

Fig. 2. Heterogeneous Pareto traces: $lift - drag$ (left), $lift - yaw$ (center), $pitch - roll$ (right)

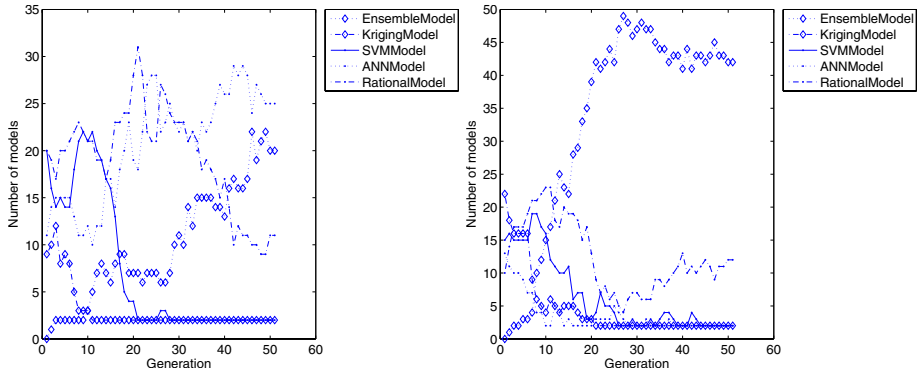

Fig. 3. Evolution of the population make-up $lift - yaw$ (left) and $pitch - yaw$ (right)

use in this case, performing much better than the other model types. Kriging and SVM perform about the same with SVM doing slightly better.

The situation in the second plot ($lift - yaw$) is different. There is a clear trade-off between both outputs. The ANN models still perform best on $lift$ but the *yaw* output is fitted most accurately by rational models and their ensembles. The difference between SVM models and Kriging models is now more marked, with the SVM models performing much better on the *yaw* output. Remark also that the distribution of the rational models is much more spread-out and chaotic than that of the other model types (which are relatively clustered). This could actually be seen for all output combinations. The reason is due to the way the order selection of the rational functions is implemented. The order selection contains too much randomness and not enough exploitation of the search space, leading to erratic results. This is currently being remedied.

The $pitch - roll$ plot has the same general structure, though the *pitch* output turns out much harder to approximate than $lift$. It is interesting to see though that ensembles turn out to be significantly superior to standalone models. It turns out that most of these ensembles turn out to be combinations of the best performing model types.

The results presented so far are of course static results. In reality the best performing model type changes as the evolution progresses. This can be seen from figure 3 which shows the relative share of each model type of the total population for two cases.

Model types that do well will have more offspring, thus increase their share. This dynamics is particularly interesting and useful in the context of active learning. It means the optimal model type changes with time, depending on how much information (data points) are available. This is exactly what one would expect (see [5] and references therein for details and examples in the single objective case).

In sum the algorithm seems to have done quite well in detecting the correlations between the outputs, and the model selection results agree with what one would expect intuitively from knowledge about the structure of the responses. The obtained accuracies are also similar to those obtained by single model type runs. It is interesting to see that, while the performance of Kriging and SVM was very similar (which is to be expected given their connection to Gaussian Process (GP) theory), the SVM results were consistently better and more robust to changes in their hyperparameters. The exact reasons for this are being investigated. It was also quite surprising to see the ensemble models to so well. In virtually all experiments the final Pareto front consisted of only of ensembles. Depending on the model selection process they are predominantly homogeneous (containing multiple models of the same type) or heterogeneous (thus 'filling in' gaps in the Pareto front).

8 Summary

The efficient identification of the best approximation method to use for a given problem is a consistently recurring question among domain experts. Particularly so in the case of multi-output problems. In this paper the authors have presented a new approach that can help tackle this problem through the use of dynamic (if active learning is enabled) multi-objective optimization with speciated evolution. This allows multiple outputs to be modeled together, giving information about the trade-off in the hyperparameter space and identification of the most suitable model type (which can be a hybrid). It further facilitates the generation of diverse ensembles (from the final Pareto front). At the same time the computational cost is roughly the same as performing multiple, independent single model type runs (though this cost is still outweighed by the simulation cost). In addition, a multi-objective evolutionary approach gives more information, naturally permits hybrid solutions (ensembles), and naturally allows the best solution to change depending on how much information (data) is available. Note that the same general approach can also be used to deal with multiple performance/accuracy criteria instead of multiple outputs.

Acknowledgments

The authors would like to thank NASA and Robert Gramacy from the University of California, Santa Cruz for providing the data and information on the LGBB example. Ivo Couckuyt is funded by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Gorissen, D., De Tommasi, L., Crombecq, K., Dhaene, T.: Sequential modeling of a low noise amplifier with neural networks and active learning. Neural Computing and Applications (accepted, 2008)
2. Pamadi, B.N., Covell, P.F., Tartabini, P.V., Murphy, K.J.: Aerodynamic characteristics and glide-back performance of langley glide-back booster. In: Proceedings of 22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island (2004)
3. Jin, Y., Sendhoff, B.: Pareto-based multiobjective machine learning: An overview and case studies. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(3), 397–415 (2008)
4. Fenicia, F., Solomatine, D.P., Savenije, H.H.G., Matgen, P.: Soft combination of local models in a multi-objective framework. Hydrology and Earth System Sciences Discussions 4(1), 91–123 (2007)
5. Gorissen, D., De Tommasi, L., Croon, J., Dhaene, T.: Automatic model type selection with heterogeneous evolution: An application to rf circuit block modeling. In: Proceedings of the IEEE Congress on Evolutionary Computation, WCCI 2008, Hong Kong (2008)
6. Fieldsend, J.E.: Multi-objective supervised learning. In: Knowles, J., Corne, D., Deb, K. (eds.) Multiobjective Problem Solving from Nature From Concepts to Applications. Natural Computing Series. LNCS. Springer, Heidelberg (2008)
7. Gorissen, D., Couckuyt, I., Dhaene, T.: Multiobjective global surrogate modeling. Technical Report TR-08-08, University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium (2008)
8. Li, X.R., Zhao, Z.: Evaluation of estimation algorithms part I: incomprehensive measures of performance. IEEE Transactions on Aerospace and Electronic Systems 42(4), 1340–1358 (2006)
9. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)
10. Voutchkov, I., Keane, A.: Multiobjective Optimization using Surrogates. In: Parmee, I. (ed.) Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture 2006, Bristol, UK, pp. 167–175 (2006)
11. Knowles, J., Nakayama, H.: Meta-modeling in multiobjective optimization. In: Multiobjective Optimization - Interactive and Evolutionary Approaches. LNCS. Springer, Heidelberg (in press, 2008)
12. Keys, A.C., Rees, L.P., Greenwood, A.G.: Performance measures for selection of metamodels to be used in simulation optimization. Decision Sciences 33, 31–58 (2007)
13. Escalante, H.J., Gomez, M.M., Sucar, L.E.: Psms for neural networks on the ijcnn 2007 agnostic vs prior knowledge challenge. In: IJCNN, pp. 678–683 (2007)
14. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby (2002)
15. Suykens, J., Gestel, T.V., Brabanter, J.D., Moor, B.D., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific Publishing Co., Pte, Ltd., Singapore (2002)