Algorithmic configuration by learning and optimization

Gabriele Iommazzo[1,2], Claudia D'Ambrosio[1], Antonio Frangioni[2], and Leo Liberti[1]

[1]CNRS LIX, École Polytechnique, Palaiseau, France
[2]Dip. di Informatica, Università di Pisa, Pisa, Italy

Abstract

We propose a methodology, based on machine learning and optimization, for selecting a solver configuration for a given instance. First, we employ a set of solved instances and configurations in order to learn a performance function of the solver. Secondly, we solve a mixed-integer nonlinear program in order to find the best algorithmic configuration based on the performance function.

1 Introduction

In this work we address the configuration of general-purpose Mathematical Programming (MP) solvers.

Most solvers have long lists of user-configurable parameters; tweaking them influences how the available algorithmic components work and how they interact with each other, and it can consequently have a significant impact on the quality of the obtained solution and/or on the efficiency of the solution process. Good solvers have effective default parameter configurations, carefully selected to provide "good" performances in most cases. Furthermore, solvers may embed heuristics that try to automatically adapt the parameter configuration to the characteristics of the instance at hand. However, default/automatic parameter configurations may still be highly suboptimal with specific instances, which require a manual search for the best parameter values. The motivation for this work lies in the fact that, due to the large amount of available parameters [10], manual tuning is highly nontrivial and time-consuming. This setting is an instance of the Algorithm Configuration Problem (ACP) [7].

Our approach for addressing the ACP on MP solvers is based on a two-fold process:

(i) in the *Performance Map Learning Phase* (PMLP), supervised Machine Learning (ML) techniques [13] are used to automatically learn a *performance function* which maps some features of the instance being solved, together with a given parameter configuration, into some measure of solver efficiency and effectiveness;

(ii) the formal properties defining the ML methodology underlying the PMLP are translated into MP terms; the resulting formulation, together with constraints encoding the compatibility of the configuration parameter values, is called the *Configuration Space Search Problem* (CSSP), a Mixed-Integer Nonlinear Program (MINLP) which, for a given instance, finds the configuration providing optimal performance w.r.t. the performance function.

The main novelty of our approach lies in the fact that we explicitly model and optimize the CSSP using the mathematical description of the ML technique used to learn the performance function. This is in contrast to most of the existing algorithmic configuration approaches, which instead employ heuristics such as experimental design methods [1], local searches [8], genetic algorithms [2], evolutionary strategies [5] and other methods [3, 12].

We remark that, differently from many other approaches, we define the performance of the target algorithm as a function of both features and controls. In this way, we account for the fact that the best configuration of a solver may vary among instances belonging to the same class of problems ("per-instance" ACP, see e.g. [3, 9]), whereas some literature works assume instead that the solution to the ACP is invariant over instances pertaining to a given problem (for example, [1, 8, 14]).

The idea of using the components of a ML predictor to define a MP formulation has been already explored and a generalisation of this research can be found, say, in [11]. The proposed approach is in fact suitable for many other applications, besides MP solvers [7], where the outcome of executing some action (e.g., run a target algorithm) depends upon the features of the problem instance and the possible action controls by the user (e.g., the parameters of the target algorithm).

2 The PMLP and the CSSP

Let $\mathcal{A}$ be the target algorithm, and:

- $\mathcal{C}_{\mathcal{A}}$ be the set of feasible configurations of $\mathcal{A}$; we assume that (a) each configuration c is a vector of binary and/or discrete values representing categorical and numerical parameters and that (b) $\mathcal{C}_{\mathcal{A}}$ can be described by means of linear constraints;
- Π be the problem to be solved, consisting of an infinite set of instances, and $\Pi' \subset \Pi$ be the (finite) set of instances used as the training set for the PMLP phase;
- F_Π be the set of features used to describe instances, encoded by vectors of continuous or discrete/categorical values (in the latter case they are labelled by reals);
- $p_{\mathcal{A}} : F_\Pi \times \mathcal{C}_{\mathcal{A}} \longrightarrow \mathbb{R}$ be the performance function (evaluated using solver performance indicators within a given time limit), which maps a pair (f, c) (instance feature, configuration) to the outcome of an execution of $\mathcal{A}$, typically in terms of the integrality gap reported by the solver, i.e. the relative discrepancy between an optimal objective function bound and the best feasible solution found so far (but other measures are possible).

2.1 Performance Map Learning Phase

In the PMLP phase we use a supervised ML predictor, e.g., Support Vector Regression (SVR), to learn the coefficient vector $\bar{\theta}$ providing the parameters of a prediction model $\bar{p}_{\mathcal{A}}(\cdot,\cdot,\theta) : F_\Pi \times \mathcal{C}_{\mathcal{A}} \to \mathbb{R}$ of the performance function $p_{\mathcal{A}}(\cdot,\cdot)$. The training set for the PMLP is

$$\mathcal{S} = \big\{(f_i, c_i, p_{\mathcal{A}}(f_i, c_i)) \mid i \in \{1 \dots s\}\big\} \;\subseteq\; F_{\Pi'} \times \mathcal{C}_{\mathcal{A}} \times \mathbb{R}, \qquad (1)$$

where $s = |\mathcal{S}|$ and the training set labels $p_{\mathcal{A}}(f_i, c_i)$ are computed on the training vectors (f_i, c_i). Our training includes a phase for determining the hyperparameters of the ML methodology by nested cross-validation [17]. We also assess the generalization error of the fully-trained predictor configured with the best hyperparameter setting.

2.2 Configuration Space Search Problem

For a given instance f and parameter vector θ, $\mathsf{CSSP}(f,\theta)$ is the problem of finding the configuration with best estimated performance $\bar{p}_{\mathcal{A}}(f,c,\theta)$:

$$\mathsf{CSSP}(f,\theta) \equiv \min_{c\in\mathcal{C}_{\mathcal{A}}} \quad \bar{p}_{\mathcal{A}}(f,c,\theta) \ . \tag{2}$$

The actual implementation of $\mathsf{CSSP}(f,\theta)$ depends on the MP formulation selected to encode $\bar{p}_{\mathcal{A}}$, which may require auxiliary variables and constraints to define the properties of the ML predictor.

If $\bar{p}_{\mathcal{A}}$ yields an accurate estimate of $p_{\mathcal{A}}$, we expect the optimum $\bar{c}$ of $\mathsf{CSSP}(f,\theta)$ to be a good approximation of the true optimal configuration c^* for solving f. However, we remark that not only $\mathsf{CSSP}(f,\theta)$ can be hard to solve, it also needs to be solved quickly (otherwise one might as well solve the instance f directly). Achieving a balance between PMLP accuracy and CSSP cost is one of the challenges of this research.

3 Experimental setup

In this paper we report results where the ML predictor of choice was SVR [16]. Its advantages are: (a) the PMLP for training an SVR can be formulated as a convex Quadratic Program (QP), which can be solved efficiently; (b) even complicated and possibly nonlinear performance functions can be learned by using the "kernel trick" [15], which reduces problematic nonlinear transformations of feature vectors to the much simpler computation of inner products; (c) the solution of the PMLP for SVR provides a closed-form algebraic expression of the performance map $\bar{p}_{\mathcal{A}}$, which allows an easier formulation of $\mathsf{CSSP}(f,\theta)$.

instance	bestFeas$_{CD}$	bestFeas$_{CSSP}$	optVal	bestRelax$_{CD}$	bestRelax$_{CSSP}$
i0001	0	0	1,193E+04	1,211E+04	**1,202E+04**
i0002	0	0	4,567E+03	4,791E+03	6,619E+06
i0003	0	0	1,155E+04	9,250E+06	1,285E+07
i0004	9,593E+03	0	1,117E+04	1,144E+04	**1,140E+04**
i0005	7,091E+03	0	8,960E+03	9,255E+03	9,322E+03
i0006	3,083E+03	3,135E+03	3,491E+03	3,609E+03	**3,597E+03**
i0007	6,921E+03	0	8,955E+03	9,060E+03	**9,028E+03**
i0008	0	0	1,539E+04	1,553E+04	**1,552E+04**
i0009	5,182E+03	0	8,778E+03	8,897E+03	**8,894E+03**
i0010	2,520E+03	0	7,721E+03	7,790E+03	1,209E+07
i0011	0	0	1,692E+04	1,705E+04	**1,704E+04**
i0012	0	0	5,691E+03	5,885E+03	5,984E+03
i0013	3,372E+03	0	3,372E+03	3,374E+03	9,527E+06
i0014	3,748E+03	0	3,832E+03	3,964E+03	1,061E+07
i0015	0	0	2,192E+03	2,449E+03	1,070E+07
i0016	2,377E+03	8,523E+02	2,382E+03	2,437E+03	**2,423E+03**
i0017	3,491E+03	0	3,781E+03	3,838E+03	4,809E+06
i0018	1,570E+02	0	6,084E+03	6,471E+03	**6,465E+03**
i0019	2,804E+03	0	4,055E+03	4,237E+03	**4,402E+03**
i0020	0	0	4,576E+03	4,646E+03	9,600E+06
i0021	0	0	4,245E+03	4,354E+03	1,343E+07

Table 1: lower and upper bounds (resp. "bestFeas" and "bestRelax") obtained using CPLEX's default configuration and the CSSP solution (resp. "*CD*" and "*CSSP*") to configure the solver

We used a Gaussian kernel during SVR training, which is the default choice in absence of any other meaningful prior [6]. This choice makes the CSSP a MINLP with a nonconvex objective function $\bar{p}_{\mathcal{A}}$. More precisely, our CSSP is:

$$\min_{c\in\mathcal{C}_{\mathcal{A}}} \sum_{i=1}^{s} \alpha_i \, e^{-\gamma\|(f_i,c_i)-(\bar{f},c)\|_2^2} \tag{3}$$

where, for all $i \leq s$, (f_i, c_i) belong to the training set, α_i are the dual solutions of the SVR, γ is the scaling parameter of the Gaussian kernel used in Eq. (3).

We tested our approach on 21 instances of the Hydro Unit Commitment problem [4]. Its purpose is to find the scheduling of a pump-storage hydro power station maximizing the revenue given by electricity selling. Each instance was encoded by 54 continuous features, representing hourly electricity prices, hourly inflows, initial and target water level of the considered reservoir. We configured 11 parameters of the IBM ILOG CPLEX MP solver [10]. Our preliminary computational experiments use CPLEX's default configuration and the CSSP solution to solve the instances. Table 1 shows that, in a number of cases, the proposed approach is capable of providing stronger relaxations than CPLEX's default. However, it still fails to find good feasible solutions, which yields overall larger integrality gaps than the solver's default. In order to tackle this issue and improve the general efficacy of the approach, we plan to test variants in the near future. We will most importantly test different features for the instances, employ alternative performance metrics for $p_{\mathcal{A}}$ and conduct experiments with other techniques than SVR.

References

[1] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using Fractional Experimental Design and Local Search. *Operations Research*, 54(1):99–114, 2006.

[2] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, pages 142–157, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] N. Belkhir, J. Dreo, P. Savant, and M. Schoenauer. Feature based algorithm configuration: A case study with differential evolution. In *PPSN XIV*, volume 9921, pages 156–165, 2016.

[4] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello. An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124, 2008.

[5] M. Brendel and M. Schoenauer. Instance-based parameter tuning for Evolutionary AI Planning. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, pages 591–598. ACM, 2011.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[7] K. Eggensperger, M. Lindauer, and F. Hutter. Pitfalls and best practices in algorithm configuration. *CoRR*, abs/1705.06058, 2017.

[8] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, 2009.

[9] F. Hutter and H. Youssef. Parameter adjustment based on performance prediction: Towards an instance-aware problem solver. Technical report, In: Technical Report: MSR-TR-2005125, Microsoft Research, (2005).

[10] IBM. *IBM ILOG CPLEX Optimization Studio CPLEX Parameters Reference*, 2014.

[11] M. Lombardi, M. Milano, and A. Bartolini. Empirical decision model learning. *Artif. Intell.*, 244:343–367, 2017.

[12] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[13] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning.* The MIT Press, 2012.

[14] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 975–980. Morgan Kaufmann Publishers Inc., 2007.

[15] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[16] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[17] S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(91), 2006.