Solver Paramter Tuning and Runtime Predictions of Flexible Hybrid Mathematical models.

Michael Barry
University of Basel / HES-SO
m.barry@unibas.ch / michael.barry@hevs.ch
Universitt Basel Petersplatz 1, 4001 Basel / Techno-Ple 1, 3960 Sierre
Switzerland

Abstract

In this research we consider the problems faced when using hybrid mathematical models to solve optimisation models. Such models can be configured to have different structures and can exert different behaviour and therefore can have a volatile search space, making runtime predictions and solver tuning a more complex problem. We propose an optimization configuration method that exploits the hybrid mathematical model structure for solver parameter tuning and runtime prediction.

Optimisation problems is active field of research and has been approached from several research communities, including the Artificial intelligence community as well as operations research community, each developing their own methods. All communities have had a steady contribution, yet most commonly compete rather than collaborate. Partly due to this rift and partially due to a difference on a conceptual level, many methods used in operation research have a distinct lack of influence from the Artificial intelligence or the wider computer science field of research despite being used heavily in industry. As a result, issues with maintenance and deployability are all to common and could be addressed by lessons learnt in the history of computer science.

Furthermore, these problems lead to highly inflexible models. As an example from industry, mathematical models of Hydro power stations are the dominant tool for planning their operation, optimising the return in profit based on the market prices (1) (5). This problem is well understood and any company operating such power stations use a mathematical model. Yet it has been noted that modifying these highly specialised models to new market environments have become difficult and many even choose to redevelop the model instead.

The inherent problem is the design of the models. Figure 1 dissects the common methods in operations research for solving an optimisation problem such as the Hydro power problem mentioned above into its separate components. The solvers are now considered to be separate from the modelling language, allowing a single model to be easily tested by different solvers. Software such as the general algebraic modelling system (GAMS) allows a user to develop a model in a single language and let it be solved by several different solvers. Similarly the input to the model is interchangeable, allowing a user to use data from, for example, a different year.

However, the model structure itself is considered to be static and never changing. More modern Hybrid models may incorporate a more flexible design and may have the potential to alter the actual behaviour of a model. For example, a more flexible Hydro power plant model could not be specific to just one existing plant, but could be flexible allowing a user to tailor it to any existing plant. This would require the model to simulate different types of turbines, each with different functions to define its behaviour. However, a model with changing behaviour presents a new problem for any methods for configuring the solver or for predicting the runtime of the solver.

Current methods in configuration focus on optimizing a solvers parameters for a specific problem. In such a case, the parameters are expected to improve the solvers efficiency in average over a selection of different input scenarios. However, if more flexible models are being developed that also change the behaviour of the model, solvers would show a more drastic change in performance for different scenarios. Therefore the configuration of the solvers parameters should take into consideration the design of the model itself, allowing the system to adjust the solvers parameters based on how the models behaviour has been configured. Similarly, the prediction of the solvers runtime becomes more complex. Changing the behaviour of the model also changes the complexity of the problem greatly, increasing both the need and the difficulty of an accurate prediction.

Therefore, the following three research questions arise.

- How can a mathematical model be implemented to allow a large degree of flexibility in it's behaviour?
- How can the parameters best be configured for a flexible and dynamic mathematical model?
- How can the runtime of a solver be accurately predicted for a flexible mathematical model?

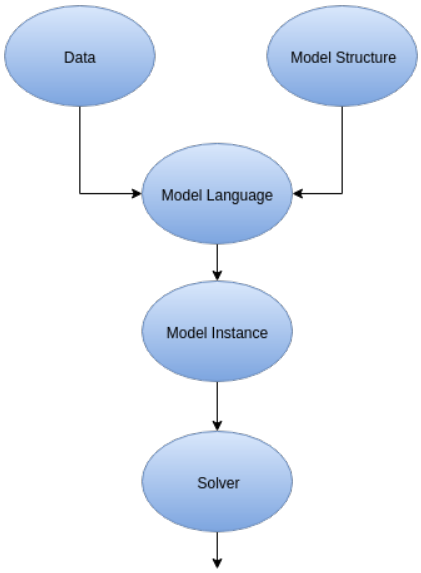

Figure 1: Graphical representation of how a problem is solved using a mathematical language and solver. Data and a model structure is input using a model language, creating a model instance. This instance can then be solved by a solver and the results are returned.

Proposed Method

Implementing software in a way that allows continues development and maintenance is well understood in computer science. One tool used to prevent it is Object Orientated design, which splits the system into several classes (module) and allows each module address another aspect of the software. Modular approaches to mathematical modelling has been used before and many new modelling languages being developed use this concept. However, this is a strong contradiction to the standard in industry, where the more classical modelling languages dominate the field. For a successful adaptation of a modular design, the concept must be usable with classical modelling languages such as GAMS. Only once a modular approach is more widely accepted will new modelling language that assists in a modular approach be more widely accepted. It also eases the integration of new models with currently used models.

A modular approach allows each aspect to be easily replaced by reimplementing the specific class. This concept can also be used in mathematical models, by separating each component into separate modules. As an example for the Hydro power plant model, it can be separated into modules each presenting an aspect of the model, such as the intraday market, a type of turbine or an environmental constraint. How this is achieved in a mathematical language is demonstrated in Figure 1 and Listing 2.

Listing 1 shows the implementation of a module containing a function describing the storage level in a Hydro power station. A further constraint on the storage variable is contained in a separate module, described in the StorageMax.gms file shown in Listing 2. By simply importing the file storageMax.gms, the module and its constraint is included in the model. A main file containing these import statements can then be used to easily modify the structure of the model by simply including or excluding modules.

Listing 1: Storage.gms

```
Equation
storage (i)
;

storage(i)..

storage(i)          =e=
 storage(i-1)
+ inflow(i)
 - release(i);

$import storageMax.gms
```

Listing 2: Storage.gms

```
Equation
storage (i)
;

storage(i)..

storage(i)          =l=          max_storage;
```

The same principle can be used for aggregate functions as shown in Listing 3 to 5 and for several versions of the same constraint as shown in Listing 6.

Listing 3: Income.gms

```
Equation
income_total (i)
;

income_total(i)..

income_total(i)          =e=
sum(m, income(i,m))
```

Listing 4: Market1.gms

```
Equation
income(i)
;

income(i)..
```

```
income(i)          =e=
price(i,1) * production(i,1)
```

Listing 5: Market2.gms

```
Equation
income(i)
;

income(i)..

income(i)          =e=
price(i,2) * production(i,2)
```

As shown in Listing 3, the total income is calculated by summing the income from each market. The income for each market is calculated in separate modules. By including or excluding the module market1 and market2, the market can be added or removed.

Listing 6: storageLoss.gms

```
Equation
storage (i)
;

storage(i)..

storage(i)          =e=
storage(i-1)
+ inflow(i)
- sum(m, release(i,m))
- loss(i)
```

Listing 6 shows an alternative implementation of the storage module described in Listing 1 and can be used to replace the previous implementation. Altogether, these methods can be used to create a model, which allows its behaviour to be modified by simple import statements.

Instances off each module can be used to model multiple occurrences of physical objects. For example, there may be different modules for different type of turbines, such as a Pelton turbine or a Francis turbine. However, a Hydro power station may contain several turbines of the same type. Therefore several instances of a turbine may exist. The concept of instantiation is fairly simple though and can be implemented in a functional language through a set of functions and a table containing the parameters for each instance. In a similar manor, it is possible for a mathematical language.

Modules depend on others and some common sense restrictions must be respected. For example, the Hydro power model must contain a type of turbine to allow power production and a market must be available to sell the energy produced. However, it is also possible to operate on several markets at the same time. Similarly, modules that implement the same aspect and therefore the same constraint but in a different may can only be selected once within a model, effectively creating an *exclusive or* relationship. Instances themselves are subject to restrictions, as a reservoir must be connected to a turbine or river to allow water to exit again to avoid flooding. The relationship between different modules and different instances can be captured in predicate logic and used as a validation method to determine a models validity before passing it to the solver.

As this approach combines a top down model, consisting of the mathematical model, and a bottom up model model, consisting of the validity logic, it would be considered a Hybrid model within the Operations Research community (4). Within the computer science community, this method is conceptually different and would be recognised as dependency injection or inversion of control, as the module with the desired behaviour is selected at runtime. Despite different ways of looking at it, such a design is understood and accepted by both communities.

The benefit of such an implementation is similar to those of Object orientated design, including reusability and maintainability. In addition, a new developer does not require knowledge of the entire system to update one aspect of the system. Furthermore, a model can then be configured by selecting which modules to use in a simulation (3) (2). This can allow a model to behave differently in each configuration as shown in Figure 2.

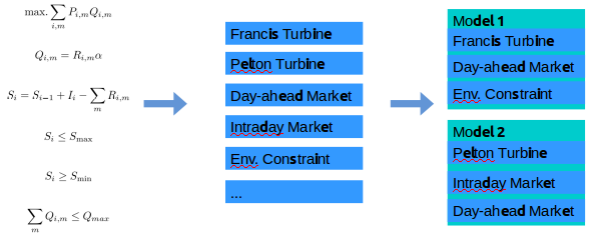

Figure 2: Graphical representation of how modules can be combined to create different configurations with different behaviours. The functions defining the model are separated into modules, which are then combined to create a model configuration.

The ability to configure the behaviour of a model sets it apart from most other model designs. Such functionality is sometime implemented to some extent utilising simple *if* statements throughout the model, such as shown in Listing 7.

Listing 7: storageLoss.gms

```
Equation
storage (i)
;

storage(i)..
```

```
storage(i)          =e=

storage(i-1)
+ inflow(i)
- sum(m, release(i,m))
- loss(i)$(include eq 1)
```

However, it is limited. Extensive use of such statements result in the model being riddled with if statements making it unreadable. In addition, each flexible aspect must be implemented manually rather than being inherently flexible by design. By using a modular approach instead it separates this logic from the functions, adding a layer of abstraction.

The resulting model would result in a large degree of flexibility, but will also result in varying complexity. Based on which and how many modules are selected at runtime, the runtime of the solver can vary greatly. Predicting the runtime of the solver can be done using machine learning methods (6). However, predicting the runtime of the solver becomes more difficult when using a model with changing behaviour. In a standard model, the search space is relatively stable compared to the proposed model. Different inputs will only modify the search space to some degree, while changing the the number of constraints and the constraints themselves changes the search space drastically.

However, there is some knowledge that can be extracted from the model structure itself and can then be used to more accurately predict the runtime. Simply using the number of modules can already help in estimating the complexity of the model as shown in Figure 3 and further information on how each module depends on the other and the validity logic contains useful knowledge of the models structure. Overall, there are many candidate features which may prove to be a good indication of the models runtime.

Using methods from machine learning such as a correlation matrix and a neural network, the best features could be selected and their relationship to the models runtime could be learned. Depending on which features are the most viable, this method could potentially also be viable for more modules or even entire models that the neural network has not been trained on. This could possibly be used as a global tool rather than being limited to a specific model. The features to be selected and their relationship to the runtime must be examined in a set of experiments. Additionally, how well the relationship can be learned and the extent to which this method can be used for unknown modules must also be investigated.

Most solvers such as the IBM CPLEX solver have several parameters that can be fine tuned by the user to boost the solvers performance for a specific problem. Parameters can modify the heuristics, probing, cutting and much more (7). Automatic configuration of these parameters has been studied to a great extent (7) and is even included in some solvers functionality. However, they do not considered a flexible model structure.

Therefore, a tuning process would have to be applied each time the structure changes. Again, knowledge from the models structure could be used to help tune the parameters for a specific structure. How much the optimal parameters for each structure vary, the performance increase and which parameters to modify must be tested in a set of experiments. Additionally, the correlation between models structure and the parameters as well as how well this relationship can be learned will have to be investigated.

The resulting system would allow a model to be configured for many different scenarios and can generate an ideal parameter configuration and runtime prediction for that scenario before being executed.

Experiments

A fully functioning prototype demonstrating the feasibility and the benefits of a model developed though a modular design will be developed. As a case study, a model of a Hydro power station is developed that has the flexibility to simulate several market and investment scenarios. Due to its flexibility, it will not be tied to a specific power plant but rather be general and can be configured for any hydro power plant. It is hoped that such a model can be used as a example for modular design and can be continuously updated and integrated with other models.

The results of the previously mentioned experiments will demonstrate how such a design can assist in predicting the runtime of a flexible model. In addition, they will determine which features of the model structure is best suited for an accurate prediction for the solvers runtime. Knowledge can be extracted from the modules themselves, such as in the most simple case the number of functions within the model, as well as their relationship with other modules, such as how many variables are shared between the modules. It is hoped that a set of features can be selected that are easily extracted from a model to allow this method be applied to modules or entire models that are not contained in the training set.

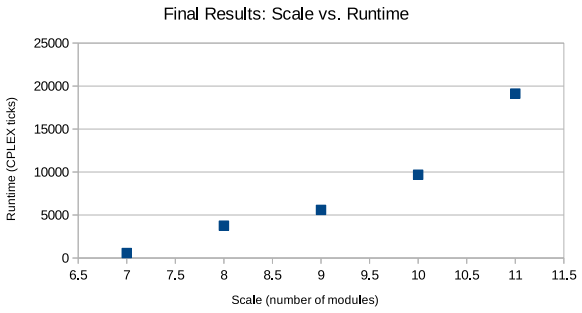

Figure 3: Initial results of running different model configurations. It shows how a simple parameter such as the number of modules has a correlation to the runtime of the model.

Therefore, it would allow this method be used on any model that follows the same design pattern. The success of this experiment relies on if general features for each module can be used, or whether the system has to learn the impact that each module has on the complexity of the model. If the latter is the case, the system would have to train on

each new module that was added before making a prediction. Although still useful, it would limit its reusability for other models.

The experiments concerning the parameter tuning of the solver will determine which parameters should be tuned for different structures to produce a performance boost. As a first step, the experiments must determine whether different configurations of the model require different parameter configurations. If this is the case, a performance boost is viable. In this case, experiments will determine which parameters are best suited for configuration and to what extent the relationship between structure configuration and optimal parameter configuration can be learned.

Evaluation

The evaluation of these methods are important in demonstrating their viability. It allows comparison to existing methods and is important for determining whether it is not only novel, but also useful to the research community.

The success of the model design will be evaluated in two case studies involving a hydro power station model. It will be compared to the current state of the art in industry and will also be evaluated on the bases of what further functionality is achieved through a modular design. The extent of its flexibility, such as whether it can simulate different type of Hydro power plants, whether it can be used to investigate investment opportunities and how well it can compete with other models in terms of accuracy in real scenarios will be tested.

The runtime prediction can be tested for accuracy by comparing the predicted runtime with actual runtime prediction. In addition, a comparison can be made with prediction methods that work with a static model structure. Although an unfair comparison, it can show the feasibility of predicting the runtime for flexible models.

The methods used for tuning the solvers parameters for different structures can be evaluated by comparing it to a base case parameter configuration. This base case can be created by choosing a configuration that is optimal in average over different model structures. Through a comparison of the solvers runtime to such a base case, the benefit of using methods based on the models structure can be measured.

Contributions

Successful investigation of the previously stated research questions would result in a fully functional system, that is adaptable, automatically configurable and predictable. However, apart from the system itself, it is expected that several scientific insights are gained. It will extend our model development methods, our knowledge of automatic parameter tuning and our understanding of what elements in a model affects its complexity.

The model design will help the efforts to transition form classical modelling methods to more agile methods similar to what is commonly used now in computer science as well as reduce the gap between the computer science and operations research communities. The model itself will be useful as an experiment base allowing the study of how small changes to a module affects its complexity. This understanding can then be used to better understand the runtime of our solvers and how best to tune the solver to minimise these effects.

Conclusions

In conclusion, a change in design of mathematical model both requires and assists in developing methods for automatic solver tuning and runtime predictions. Current methods are limited by the assumption that the structure remains static and the search space only shifts due to different inputs, rather than different model behaviours. Changing behaviours makes the tuning of the solver and the prediction of its performance more difficult, but may also deepen our understanding of this relationship, further improving the state of the art.

Acknowledgements

This work has been done in the context of the SNSF funded project *Hydro Power Operation and Economic Performance in a Changing Market Environment*. The project is part of the National Research Programme *Energy Transition* (NRP70).

References

[1] Alfieri, L.; Perona, P.; and Burlando, P. 2006. Optimal water allocation for an alpine hydropower system under changing scenarios. *Water resources management* 20(5):761–778.

[2] Barry, M., and Schumann, R. 2015. Dynamic and configurable mathematical modelling of a hydropower plant research in progress paper. In *Presented at the 29. Workshop "Planen, Scheduling und Konfigurieren, Entwerfen" (PuK 2015)*.

[3] Barry, M.; Schillinger, M.; Weigt, H.; and Schumann, R. 2015. Configuration of hydro power plant mathematical models. In *Energy Informatics: Proceedings of the Energieinformatik 2015*, volume 9424 of *Lecture Notes in Computer Sciences*. Springer.

[4] Böhringer, C. 1998. The synthesis of bottom-up and top-down in energy policy modeling. *Energy economics* 20(3):233–248.

[5] Guo, S.; Chen, J.; Li, Y.; Liu, P.; and Li, T. 2011. Joint operation of the multi-reservoir system of the three gorges and the qingjiang cascade reservoirs. *Energies* 4(7):1036–1050.

[6] Hutter, F.; Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2014. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence* 206:79–111.

[7] Klotz, E., and Newman, A. M. 2013. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science* 18(1):18–32.