Simulation and Optimization as Modular Tasks within a Framework on High-Performance-Computing-Platforms

Jochen Wittmann[1], Kai Himstedt[2], Steven Köhler[2], Dietmar P. F. Möller[2,3]

Abstract

In the context of the project „Effizienter Flughafen 2030“ separate, autarkic models simulating the different processes at an airport have been coupled to build an integrative over-all model. For scenario analysis and to identify the relevant model parameters for the over-all behavior it was postulated to fit input parameter values automatically with regard to a certain optimal behavior of the coupled system. To solve this task, a framework approach had been developed using soft-computing optimization methods. Independently of its original motivation and application, this approach can be applied to all complex optimization problems, if an encapsulated specification of the model components and the optimization is provided. The soft computing approach has the advantage to be able to deal easily with typical phenomena known in the area of environmental modeling such as lack of definition, non-linearity, or incomplete data. On the technical level it was a quite natural step to transfer the concept of the model coupling to the complete framework including the optimization component and to implement the complete functionality as a web service. For the prototype introduced in this paper genetic algorithms are selected for the optimization module. The architecture of the genetic solver demonstrates the general framework approach and is able to use high-performance-computing platforms to produce considerable speedups without demanding high bandwidths or low latency interconnections for the platform used.

A simple application example will demonstrate the interfaces for the model and the genetic solver that have to be provided to encapsulate these components for usage in the framework. The outlook deals with the optimization of the parameters of the genetic solver itself and proposes to interpret the black-box web-service interface of the solver as a model itself such cascading two genetic solvers: the inner one to optimize the simulation model parameters and the outer one to optimize the genetic algorithm parameters of the inner one.

1. Introduction

One of the main task for a modelling and simulation approach is to optimize the behaviour of the system under observation. Therefore, the relation between the two modules “simulation model” and “optimization algorithm” is well developed since a long time (see e.g. [19]). Mostly however, there will be problems with the calculation effort to solve the coupled simulation-optimization problem. In general, the simulation model will have input parameters of different types (int, float, bool, double, …) to vary for the simulation runs. The simulation-based optimization has to find a set of values for these inputs in regard to a given objective function. This function evaluates the simulation run in a certain way and delivers a measure for the quality of the current set of input parameter values.

[1] HTW Berlin, FB2, Umweltinformatik, Wilhelminenhofstr. 75A, 12459 Berlin, wittmann@htw-berlin.de

[2] Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, [himstedt|skoehler|dmoeller]@informatik.uni-hamburg.de

[3] TU Clausthal, Institut für Angewandte Stochastik und Operations Research, Erzstr. 1, 38678 Clausthal-Zellerfeld, dietmar.moeller@tu-clausthal.de

A very simple optimization problem might be to find optimal values for two integer parameters in the interval of 0 to 19. The first approach to solve the problem would be a parameter variation algorithm that finds all possible combinations, runs for all these (400) input vectors the model, calculates the objective function, and thus determines the optimum by a "brute force"-approach. The problem for practical use is that the number of combinations grows up exponentially with the number of input parameters. If we increase the number of inputs in this example from 2 to 4 there would be 160000 runs be necessary instead of the 400 runs for 2 inputs.

For real problems more efficient search strategies are used. Often used are the tree-search algorithms on the one hand, and the so called genetic algorithms on the other hand. For a typical example where both optimization strategies are applied successfully and compared in a realistic air traffic context see [7].

In this paper, the focus will be on the genetic algorithms because of an already existing implementation of this strategy type in connection with the framework introduced. Using these algorithms, the problem of the exponential relation between the number of inputs and the number of combination between them (all representing an individual simulation run) still exists, but the setting of the problem allows an easy parallelization of the independent simulation runs necessary and thus, nevertheless, leads to feasible runtimes. However, the technical problem how to manage this parallelization task still remains and that is exactly the initial situation for this paper.

Thus, a short introduction how to increase the computing power by high performance systems will be given in the next section. Then, the architecture of the optimization framework is introduced and will be exemplified by a textbook-example.

2. High-Performance-Computing and Cloud-Computing

We will not focus on a distinctive definition of high-performance-computing in this section, because the value of "high performance" always will depend on the current state of the technique and available hardware platforms. But one main characteristic of all the approaches is the support for parallelization. First a symmetrical multi-processor (SMP) system could be used that typically offers a high communication bandwidth and supports shared-memory access. However, the number of processors for such a symmetrical multiprocessor system is limited.

The further extension of such a SMP system are cluster systems with cluster nodes consisting of SMPs. The necessary communication between the nodes may be realized e.g. by a message passing interface (see [10, 15-16]) if there is a demand for fine granular communication and/or synchronisation. In comparison to SMP systems the number of processors is increased, but in return, fast shared-memory access will be restricted on the level of the cluster nodes.

One currently developed type of high-performance computing is the so-called "cloud-computing" allowing to rent via the internet the amount of computing power required and for exactly the time interval needed to solve a problem. A typical example for such a cloud environment is the Google Compute Engine [6]. An alternative to the offer of Google [9] is Amazon [1] with the Amazon Elastic Compute Cloud (EC2) and the Amazon Web Services. To give an impression of the pricing for such services: the costs for a machine with 8 virtual CPUs – comparable to 8 Hardware-Hyper-Threads of a current Intel-Server-CPU – are about 0.49 € per hour at Amazon [2]. The administration of the virtual cluster is managed by a browser-based management console, the access is realized by remote-desktop clients (for Windows) or ssh (for Linux). Additionally, there are Graphics Processing Units (GPU) offered and the user has the possibility to configure his cloud dynamically according to the currently needed load profile.

Thus, the main advantages of cloud computing lie in zero efforts for maintaining the hardware and the option to rent exactly the amount of computing power needed. This however is exactly the typical situation for every modelling and simulation study: after a long period of model development on standard PCs comes a phase of intense experimentation for parameter variation and/or optimization with numberless simulation runs and strong demands concerning computing power. How the simulation has to be adapted to benefit from the offer of a compute cloud will be shown in the next section.

3. Concept of the Optimization-Framework

As already mentioned, one of the main tasks during a simulation study is optimization, i.e. the fitting of free parameters within the model description to the model data or the search for an optimal behaviour of the modelled system during the simulation period. It is nearly impossible to apply analytical methods for optimization because

- the system generally has a large number of free parameters, and
- there is no analytical and explicit junction between the free parameters and the value of the objective function. This is the reason for the use of simulation: the simulation model itself represents the connection between input values and resulting value of the objective function and therefore has to be evaluated for each setting of the inputs individually.

The use of optimization methods and simulation models should be structured as independent as possible to fulfil the postulation of encapsulation and modularisation. Fig. 1 shows the general concept and the architecture of the optimization framework in some more detail.

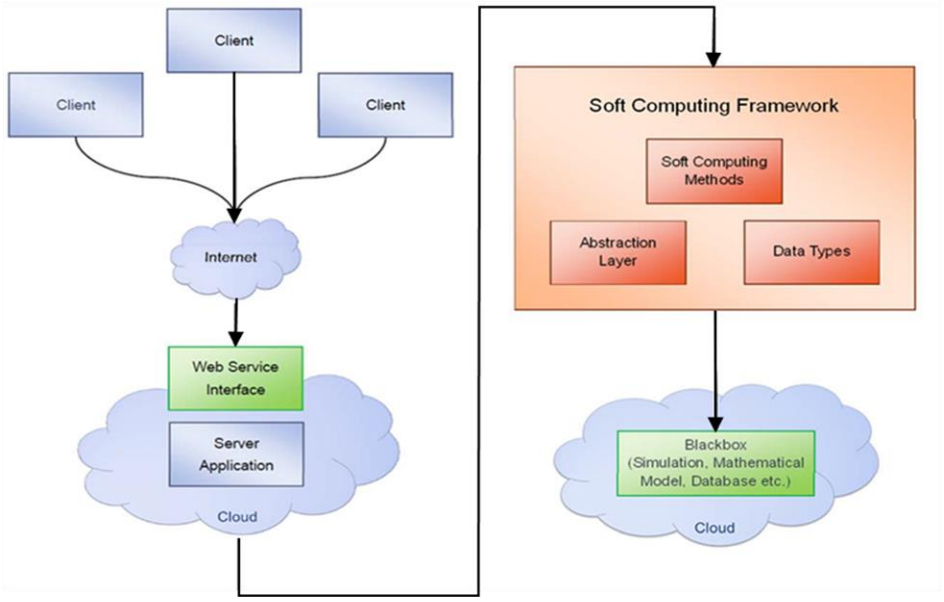

Fig 1. Overview over the Framework-Architecture

The special feature of the framework proposed is that there is no need to know anything about the simulation model or the mathematical model. The system to optimize appears as a real black-box activated by an easy to define web-service-interface. This allows flexible hosting of the simulation models in the cloud.

4. Prototype

The basis of the implementation are Web Services (WS) [21] and the use of XML-standards for exchange of structured data.

Fig. 2 shows the architecture of the genetic solver as one implementation of the optimization framework concept. The input parameters of the client (upper left) are transformed to internal structures (WSF Staff-Framework [23] and possible alternatives, see in [12]). In the middle the core of the optimization is figured, calling the simulation to determine the objective function for the different chromosomes in parallel and distributed as web-service requests by the load balancer (lower part of the figure).

The main benefit of the architecture proposed lies in the fact how the interfaces between these different levels are designed and especially in their independency from a specific simulation model and the optimization algorithm used. The following example will demonstrate, what a user of the framework has to specify to get the whole system running.

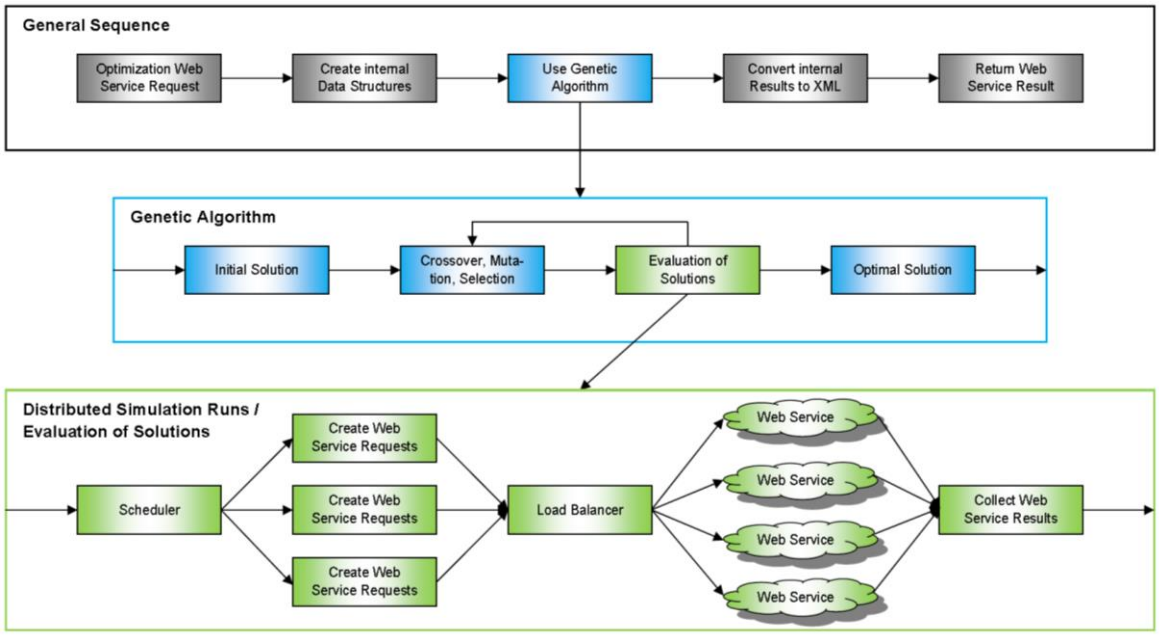

Fig. 2. Architecture of the Genetic Solver

5. An Example

To show the elements of the framework and the additional work for the user to encapsulate the relevant elements, a very simple textbook example is implemented: Fig. 3 shows the problem dealing with the production numbers of two products under the restriction of machine capacities (from [18]).

The objective function to maximize is given by the earnings for the two products without exceeding the capacities of the machines. Concerning the interface, the inputs are the numbers for A and B to produce and the machine capacities. The output is the value for the overall profit.

The framework architecture demands from the user to transform this interface as follows:

1. Defining the interface as a web-service interface (see Fig. 4)
   The simulation model is transformed to a black-box-component by the web-service interface. The model could be of arbitrary complexity that is hidden behind the input-interface and the encapsulation. For the example, the model just calculates the over-all earnings and represents just a placeholder for the general functionality.

2. Implementing the simulation as a web service (see Fig. 5)
   Here an implementation by C++ using the WSF Staff-Framework is used. The total earnings are calculated and returned; in case of a violation of the restrictions the earnings are cut to 0.

3. Configuring the optimization algorithm as a web service (see Fig. 6a)
   Here the solver interface is depicted on the highest level of its XML Schema description [22]. By the *SolverConfigurationType* the typical parameters are set. Of special interest is the entry *SimulationServiceURLs* that indicates a list of servers in the compute cloud providing identical copies of the simulation model and thus being prepared for a parallel execution of several runs.

4. Configuring the simulation as a web service (see Fig. 6b)
   The configuration of the simulation execution defines the name of the simulation function and kind of objective evaluation selected (e.g. minimize or maximize value). In addition, the input- and the output parameters of the simulation model are specified. Even structured parameter types may be used as it is natural for XML and Web Services.

5. Formatting the output of the optimization algorithm (see Fig. 7)
   The result of the solver is given here as a typical XML formatted output extracted from a protocol. The XML-representation of the output contains all the information given in the *SimulationConfiguration* when calling the web service, such as *isModifiable* for the free parameters. To simplify and by the hope to shorten the optimization the range of parameters of this type may be restricted by the user (e.g. *countA* may not exceed a value of 150 (limited by max. capacity of Machine II), and *countB* may not exceed a value of 60 (limited by max. capacity of Machine III).

	A	B	C	D	E
1	**Maximizing the Profit (considering a 1 month period)**				
2					
3		**Product**	**Product**		
4		**A**	**B**		
5	**Sales Price**	**1,000 €**	**3,000 €**		
6	**Production Costs**	**700 €**	**2,500 €**		
7	**Profit per Unit**	**300 €**	**500 €**		
8					
9					
10	**Required Machine Hours per Product**			**Capacities Used**	**Max. Capacities Available**
11	**Machine I**	**1**	**2**	**170**	**170**
12	**Machine II**	**1**	**1**	**110**	**150**
13	**Machine III**	**0**	**3**	**180**	**180**
14					
15	**Production Numbers**	**50**	**60**		
16	**Profit per Product**	**15,000 €**	**30,000 €**		
17					
18	**Overall Profit**			**45,000 €**	**<- Optimum?**

Fig. 3. The Textbook Optimization Problem

Web Service Request

```
<soapenv:Body>
   <calculateProfit>
      <countA>50</countA>
      <countB>60</countB>
   <calculateProfit>
</soapenv:Body>
```

```
<soapenv:Body>
   <calculateProfitResult>45000</calculateProfitResult>
</soapenv:Body>
```

Web Service Response

Fig. 4. Encapsulating the Simulation by the Web-Service Interface

```
int MyExcelSimulationImpl::calculateProfit(int countA, int countB)
{
// limits machine I - III
    if (countA + 2 * countB > 170) return 0;
    if (countA + countB > 150) return 0;
    if (3 * countB > 180) return 0;

    return 300 * countA + 500 * countB;
}
```

Fig. 5. Implementation of the Simulation Model as a Web Service in C++

```
<complexType name="SolverConfigurationType">
  <sequence>
    <element name="mutationProbability" type="double"></element>
    <element name="crossoverProbability" type="double"></element>
    <element name="competitors" type="int"></element>
    <element name="populationSize" type="int"></element>
    <element name="generations" type="int"></element>
    <element name="SimulationServiceURLs" type="URLsType"></element>
  </sequence>
</complexType>
```

Fig. 6a. Solver Web Service (XML Schema Fragment for Configuration of the Genetic Algorithm)

```
<complexType name="SimulationConfigurationType">
  <sequence>
    <element name="name" type="string"></element>
    <element name="simulationFunction" type="string"></element>
    <element name="evaluationFunction" type="string"></element>
    <element name="inputParameters" type="ContainerType"></element>
    <element name="outputParameters" type="ContainerType"></element>
  </sequence>
</complexType>
```

Fig. 6b. Solver Web Service (XML Schema Fragment for Configuration of the Simulation)

```
<result>
  <wsFunction name="calculateProfit">
    <parameters>
      <int32 name="countA" isModifiable="true" minValue="0" maxValue="150">
        130
      </int32>
      <int32 name="countB" isModifiable="true" minValue="0" maxValue="60">
        20
      </int32>
    </parameters>
    <resultType>
      <int32 name="calculateProfitResult">
        49000
      </int32>
    </resultType>
  </wsFunction>
</result>
```

```
Solver Configuration:
mutationProbability:  0.05
crossoverProbability:  0.2
competitors:  3
populationSize:  500
generations:  10
```

Fig. 7. Output (Example) of the Genetic Solver

As one can see, all the interface tasks are managed in just one page of code. The code segments are easily understandable and (especially if they are marked for the unexperienced user as blocks to edit within the whole code file) easily to fill, too. The user just has to specify the objects of the interfaces of simulation and solver such as parameters, restrictions, and paths at the position indicated within the prepared code blocks. This is just the specification level the user is acquainted with.

6. Conclusion and Outlook

Based on the assumption of the need for high-performance computing for optimization tasks in connection with simulation studies, the paper proposes a framework-approach for platforms like the Amazon Elastic Compute Cloud. The example using genetic algorithms as optimization solver and an encapsulated simulation based on a textbook problem shows the feasibility and the level and design of a possible user interface. It shows, that this approach is well suited to provide a high-performance-solution as „Simulation Computing as a Service“ in analogy to the so called „Everything as a service“ [3] with low level effort for implementation and low investment costs for hardware.

The use of genetic algorithms seems to be ideally suited for the modularisation because

1. there is no internal knowledge about specials of the model and/or simulation necessary and the interface can be reduced on the typed input-/output-parameter set, and
2. the search can be easily parallelized by parallel simulation runs each of them evaluating the fitness of one chromosome under observation.

On the side of simulation, models in C, C++, Java, and especially MatLab-models can be encapsulated as shown for the example and realized as web-services. There are already implementations running for these types of model implementations. Any other simulation system may be integrated as well if it can be controlled externally by an Application Programming Interface (API).

References

[1] Amazon, *Amazon Elastic Compute Cloud (Amazon EC2)*, ws.amazon.com/de/ec2/ , March 21, 2014.

[2] Amazon, *Amazon EC2 – Preise*, aws.amazon.com/de/ec2/pricing , March 21, 2014.

[3] Banerjee, P., Bash, C., Friedrich, R., Goldsack, P., Huberman, B. A., Manley, J., Patel, C., Ranganathan, and P., Veitch, A., “Everything as a Service: Powering the New Information Economy” In *IEEE Computer*, vol. 44, pp. 36-43, 2011.

[4] BMBF, *Luftfahrtcluster Metropolregion Hamburg*, www.2012.hightech-strategie.de/de/1533.php , March 20, 2014.

[5] Brain, Z., and Addicoat, M., "Using Meta-Genetic Algorithms to tune parameters of Genetic Algorithms to find lowest energy Molecular Conformers" In *Proceedings of the Alife XII. Conference Odense, Denmark*, pp. 378-385, 2010.

[6] ExtremeTech, *Google Compute Engine*, www.extremetech.com/extreme/131962-google-compute-engine-for-2-millionday-your-company-can-run-the-third-fastest-supercomputer-in-the-world , March 22, 2014.

[7] Gianazza, D., and Alliot, J.-M., "Optimization of Air Traffic Control Sector Configurations using Tree Search Methods and Genetic Algorithms" In *Proceedings of the Digital Avionics Systems Conference 2002*, pp. 2.A.5-1 - 2.A.5-8, 2002.

[8] GBU, *ProModel – Simulationsprogramme für Produktion und Logistik*, www.gbumbh.de/html/promodel.htm , March 22, 2014.

[9] Google, *Google Cloud Platform - Compute Engine. Run large-scale workloads on virtual machines hosted on Google's infrastructure*, cloud.google.com/products/compute-engine/ , March 19, 2014.

[10] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI – Portable Parallel Programming with the Message-Passing Interface*, The MIT Press. Cambridge, London 1994.

[11] Himstedt, K., Wittmann, J., and Möller, D. P. F., "Modellkopplung und Szenarioanalyse am Beispiel des Projekts ‚Effizienter Flughafen 2030' " In Wittmann, J., and Maretis, D. K. (eds.) *Simulation in Umwelt- und Geowissenschaften, Workshop Osnabrück 2010*, Shaker Verlag, Aachen, pp. 91-104, 2010.

[12] Köhler, S., Himstedt, K., and Möller, D. P. F., "Konzeptueller Ansatz zur Anbindung eines Soft Computing Frameworks an eine Web Service basierte Simulationsumgebung" In Wittmann, J., and Page, B. (eds.) *Simulation in Umwelt- und Geowissenschaften, Workshop Hamburg 2012*, Shaker Verlag, Aachen, pp. 143-156, 2012.

[13] Köhler, S., Himstedt, K., and Möller, D. P. F., "A Conceptual Approach for a Soft Computing Framework to Determine Correlations in High-Dimensional Data" In *7. MATHMOD Wien 2012*, IFAC Online, 5 pages, 2012.

[14] MathWorks, *MATLAB – Die Sprache für technische Berechnungen*, www.mathworks.de/products/matlab , March 21, 2014.

[15] MPI, *Message Passing Interface (MPI) Forum Home Page*, http://www.mpi-forum.org/ , March 21, 2014.

[16] MPI, *MPICH – High-Performance Portable MPI*, http://www.mpich.org , March 20, 2014.

[17] Russel, S. J., and Norvig, P., *Artificial Intelligence: A Modern Approach*, Second Edition, Prentice Hall – Pearson Education Inc. Upper Saddle River, New Jersey, 2003.

[18] Schmidheiny, K., and Wälti, M., "Doing Economics with the Computer" In *Einführung in die Wirtschaftsinformatik*, Institut für Wirtschaftsinformatik, Abteilung Informationsmanagement, Universität Bern, 5 pages, 2002.

[19] Wittmann, J., "The Linkage of Simulation and Optimization by the SIMPLEX II Experiment Description Language" In Krug, W., and Lehmann, A. (eds.), *Proceedings of the 1992 European Simulation Symposium*, Dresden, SCS, San Diego, 1992.

[20] Wittmann, J., Himstedt, K., and Möller, D. P. F., "Ein Pipeliningkonzept zur Modellierung der Passagier- und Gepäckbearbeitung am Flughafen" In Gnauck, A., and Luther, B. (eds.), *ASIM 2009 – 20. Symposium Simulationstechnik in Cottbus*, Shaker Verlag, Aachen, pp. 404-414, 2009.

[21] W3C, *Web Services Activity*, http://www.w3.org/2002/ws , March 21, 2014.

[22] W3C, *XML Schema*, http://www.w3.org/XML/Schema , March 21, 2014.

[23] WSF Staff, *staff – Open source Web Service Framework for C++ based on Axis2/C*, https://code.google.com/p/staff/ , March 21, 2014

[24] Zadeh, L. A., "Fuzzy Logic, Neural Networks, and Soft Computing" *Communication of the ACM*, vol. 37 pp. 77-84, 1994.