Pumps + Systems
Compressors, Compressed Air and Vacuum Technology

SHEP: An Innovative Language to create and evaluate Optimization Programs for Pump Systems

Author:

Benjamin Saul
Research Assistant
Martin Luther University Halle-Wittenberg, Institute of Computer Science
06120 Halle (Saale), Germany

Co-Author:

Dipl.-Ing. Philipp Pöttgen
Research Assistant
Technische Universität Darmstadt, Fluidsystemtechnik
64287 Darmstadt, Germany

Co-Author:

Prof. Dr. rer. nat. habil. Wolf Zimmermann
University Professor
Martin Luther University Halle-Wittenberg, Institute of Computer Science
06120 Halle (Saale), Germany

Co-Author:

Prof. Dr.-Ing. Peter F. Pelz
University Professor
Technische Universität Darmstadt, Fluidsystemtechnik
64287 Darmstadt, Germany

VDMA
German Engineering Federation

Pumps + Systems
Chairman: Dr. Sönke Brodersen
Compressors, Compressed Air and Vacuum Technology
Chairman: Alexander W. Peters

Lyoner Str. 18
60528 Frankfurt am Main, Germany
Telefon +49 69 66 03-12 82
Telefax +49 69 66 03-22 82
E-Mail pu@vdma.org
Internet pu.vdma.org

VDMA
Technology Serving People

Summary

Planning a pump system is a difficult task. Many different load profiles (scenarios) are given and have to be supported by the system. You can choose from a wide range of pumps and fittings to include them in the system. Many configurations of the chosen components have to be considered for this task. Therefore, an exponential number of possible systems can be imagined, more than a system designer can all think of. From this set of possible systems, it is hard to find the optimal system with minimal power consumption, acquisition costs or maintenance costs.

Partners of this project developed a mathematical optimization program to compute this optimal pump system [1]. But such an optimization program is hard to read and to understand.

Therefore, a domain specific language was designed to generate this optimization program. The language includes among others pump specifications, a selection of components to be used, their connections and different load profiles. The generated program is then solved by external software tools and evaluated to show the optimal system in a simulation model. The generated optimization program is analyzed to achieve user friendly feedback about feasibility of a system with the specific requirements.

This paper presents some features of the language to proof that it is possible to specify a pumping system optimization scenario in a readable manner. Pumps are described by their characteristics, connection types and their costs. Characteristics are automatically linearized for a detailed model of the available operation points of the pump.

If it is not possible to configure a system based on the given constraints, an error report will be generated indicating the problem in the original specification. For example, there would be an error message if the required pressure cannot be achieved by only using the given pumps.

Another approach of this work is to evaluate the results of the optimization program solver. The result of the solver is a set of variables with their values. From this information a simulation model next to a layout of the cost optimal pump system are generated.

In summary this work makes the power of mathematical optimization methods usable for a wide range of users. It is easy to plan an energy optimal system with the presented tool which supports all requirements. Compiler technologies can analyze the planned system and give the user a readable feedback for his work. Also it is possible to generate faster solvable optimization programs than normal crafted programs.

Introduction

Pump systems cause more than a tenth of the European energy consumption [2]. This consumption can be minimized by considering energy efficiency at the design of a pump system, i.e. choosing specific pumps, adjusting their rotational speed and configuring the overall system layout.

A common design task is to transport a fluid from sinks to sources with respect to flow and pressure. Pumps, pipes and valves are used to meet these requirements. Designing a pump system which meets specific load profiles and available building space and components requires considering many possible configurations. For example, one could add additional pumps for load sharing. The question whether or not a component is used in the system leads to an exponential number of configurations, from which the system architect has to choose the best one possible. Figure 1 shows an example system with optional components to choose from.

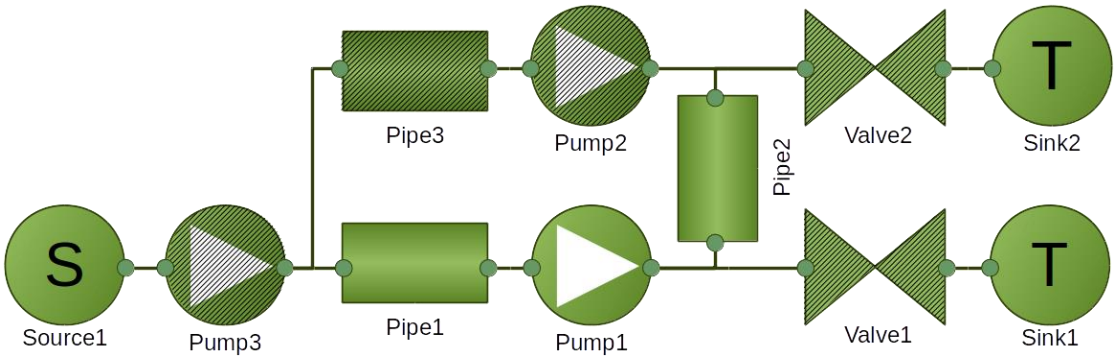


Figure 1: Example for a pumping system. Hatched components are optional and do not have to be included in the optimal system.

An optimal configuration with respect to energy consumption or other optimization criteria, which meets all requirements, can be computed by solving a Mixed-Integer Linear Problem (MILP). An MILP is a set of equations and constraints which represents component and system behavior like flow preservation and pressure increase. An application of this method can be found in [3]. Formulating this MILP requires knowledge of mathematical methods and is therefore not suitable for system architects.

A domain specific language and workflow were developed making these optimization methods usable for pump system engineers. The language is supposed to be a shorter and clearer way to describe possible pumping systems than the mathematical formulation. It is a textual language whose specification needs no further software than a standard editor.

The following sections show the workflow for designing an optimal pump system and SHEP, the language developed for this purpose. The language is shown by a pump model and a complete system specification of an example system. The final section concludes this work and shows possible future applications.

Designing Optimal Pump Systems

The proposed workflow consists of the methods presented in [3] but is expanded for better usability. Figure 2 shows this approach.

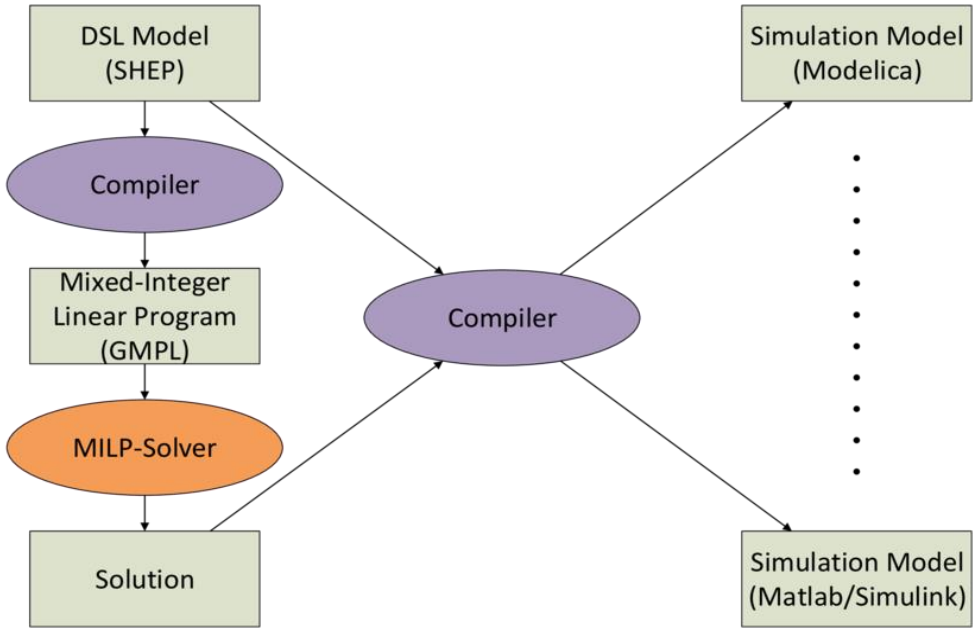


Figure 2: Workflow for computing optimal pumping systems.

The system architect specifies available component models like pumps and valves in the domain specific language. These are the component groups which are available to be used in the pump system. Component models can be stored in a common file based database and referenced in the model. Using these components possible system topologies have to be specified, giving the optimization process a set of configurations to choose from. Finally, load profiles can be modelled as numerical constraints for pressure and flow.

From this specification, the compiler, a specific software program which translates text based models in other models, generates the optimization program, which is then solved by a standard MILP-Solver. The compiler was developed by the *Eli Toolkit* [4]. In a second run the compiler processes the results to form a simulation model in *Modelica* or other formats which the system architect can use to inspect details of the optimal system.

The Pump Model in SHEP

Each pump model identifies to a family of behavior equivalent components. Describing a pump includes its working behavior like load and energy consumption, the occurring costs and incoming and outgoing pipe connectors.

The behavior of a pump is defined by its characteristics. It defines the possible flows and pressures a pump can support, as well as the related energy consumption. Pump characteristics are commonly stored as a set of measurement points. An example characteristic is given in Figure 3.

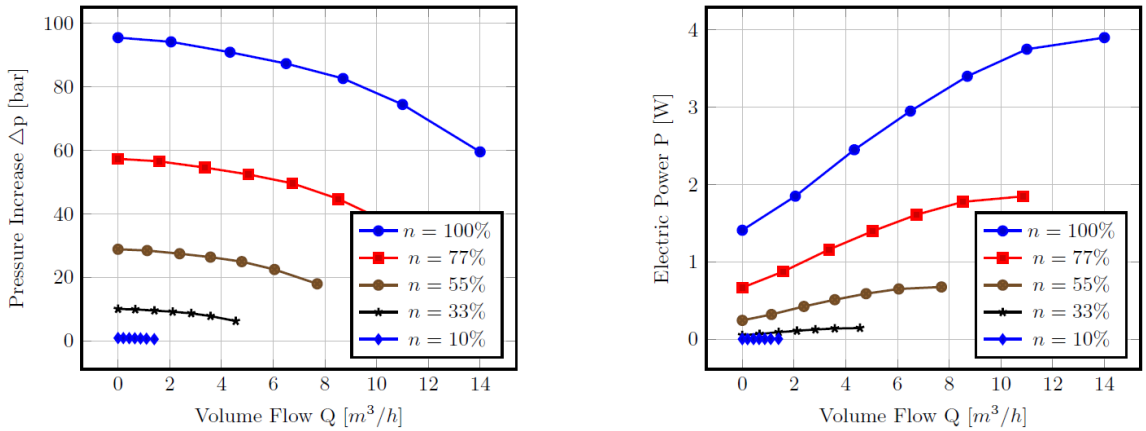


Figure 3: Characteristic of an example pump.

The occurring costs are costs for purchase and installation of the components as well for long term support and maintenance. The incoming and outgoing pipe connectors restrict possible configurations with other components, they also specify the highest supported pressure at the connection. Listing 1 gives a description of an example pump.

```
pump Movitec
  characteristic
    speed    head    flow    power;
    10.000   0.955   0.000 0.002;
    10.000   0.596   1.400 0.005;
    32.500 10.091   0.000 0.053;
    32.500   6.292   4.550 0.146;
   // ...
  costs
    purchase = 1800;
  ports
    in Flange;
    out Flange;
end
```

Listing 1: Domain specific description for the pump Movitec.

The model is introduced by the keyword **pump** and closed by the keyword **end**. The three parts of the specification match to the topics mentioned above. The characteristic is given as set of points. Units must be conform to the specification. The **ports** have the type Flange and can therefore only be connected to other Flange connectors.

From this specification, the compiler generates an optimization code. This code includes common equations, like flow preservation and pressure increase, but also considers the specific behavior of each pump by calculating the optimization model of the characteristic. An excerpt from the pump model in the optimization language is given in Listing 2 to demonstrate the saved lines of code.

```
set Movitec within pumps;

set   Movitec_fh dimen 2; # flow, head
param Movitec_speed {basics_Movitec};
param Movitec_power {basics_Movitec};

param cntSimplizes_Movitec default 0;
set simplizes_Movitec := 0..cntSimplizes_Movitec - 1;

set triangulation_Movitec within basics_Movitec cross simplizes_Movitec;

var l_Movitec {scenarios cross Movitec cross Movitec_fh}, >=0, <=1;
var simplex_Movitec {scenarios cross Movitec cross simplizes_Movitec}, binary;

subject to # common equations for all pumps

FlowPreservation {S in scenarios, C in pumps}:
  _flow_port[S,C,'_in'] = - _flow_port[S,C,'_out'];

PressureIncrease {S in scenarios, C in pumps}:
  _pressure_port[S,C,'_out'] = _pressure_port[S,C,'_in'] + head[S,C];

# equations for linearization of the pump Movitec
CharaLinACC1_Movitec { S in scenarios, P in _Movitec}:
  sum {simplex in simplizes_Movitec} simplex_Movitec[S, P, simplex] = isPowered[S, P];

CharaLin_isPowered_Movitec{S in scenarios, P in Movitec}:
  isPowered[S,P] = sum {(f_ss, h_ss) in Movitec_fh} l_Movitec[S, P, f_ss, h_ss];

CharaLin_speed_Movitec {S in scenarios, P in Movitec}:
  speed[S,P]
    = sum {(f_ss, h_ss) in Movitec_fh} l_Movitec[S,P,f_ss,h_ss] * Movitec_speed[f_ss,h_ss];

data; # interpolation data
param: Movitec_fh: # characteristic
Movitec_speed Movitec_power :=
0.000   0.955  10.000         0.002
1.400   0.596  10.000         0.005
0.000  10.091  32.500         0.053
# ...

param cntSimplizes_Movitec # triangulation
:= 8;
set triangulation_Movitec :=
(0, 0.955, *)      1 0
(1.4, 0.596, *)    1
(0, 10.091, *)     3 2 0
# ...
```

Listing 2: Excerpt from the generated equations for the pump model Movitec (Listing 1).

Modelling the characteristic requires a method to compute every possible operating point. In this work, an interpolation method is used, presented in [5]. To interpolate the characteristic, the measurement points are triangulated as shown in Figure 4.

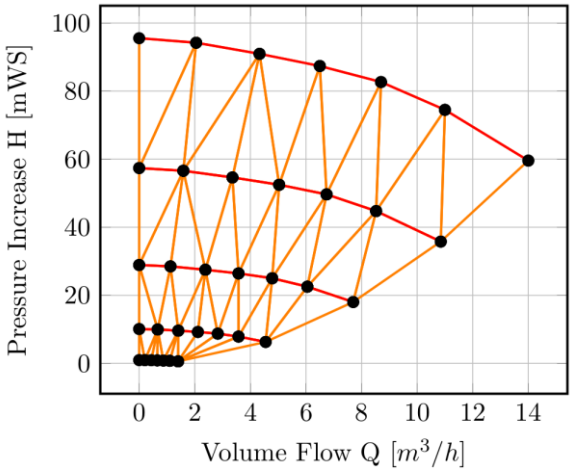


Figure 4: Triangulation of a characteristic.

Modelling a Pump System

Modelling a pumping system requires three parts: the components available to design the system, their connections as well as scenarios. Scenarios are possible states or load profiles the system must support, like daytime and nighttime or different seasons. Weights approximate the amount of time spent within a state and define the amount of energy used in the future system. Listing 3 represents the system from Figure 1.

A specification of components and their topology should offer the possibility to restrict solutions. These may represent real world facts, such as available components or limited space. There can be any number of components defined in a system. No restrictions regarding the connection of components except for compatibility constraints (type, diameter) exist. Fittings, i.e. tee or cross fittings, are currently neglected in the domain specific model but could be generated based on the solution.

In the example, next to the previous specified Movitec pump, also a CommonPipe and a StandardValve appear. They were specified analogously to the Movitec pump. Sink and Source are abstract components who represent input and output of the system.

The keyword **optional** denotes components which are not required in an optimal solution, whereas a component without this annotation must be used in any design, especially the optimal design. Different sub sets of optional components, such as pumps and pipes, may be used to form a valid solution.

For simplicity Listing 3 only contains one scenario. A scenario represents the requirements and given attributes of a system for a specific load profile, i.e. day and night profile. The weight is the expected duration for the scenario.

The compiler checks the specification for common errors. For example, an error occurs when the user tries to connect components with different port types. Another source for errors is the use of pumps which could not perform the required pressure increase. These pumps must be marked in the domain specific model because they lead to an infeasible optimization problem, which is more complex to analyze.

```
system MySys
  Source       Source1;
  Movitec      Pump1, Pump2 optional, Pump3 optional;
  CommonPipe   Pipe1, Pipe2, Pipe3 optional;
  StandardValve Valve1 optional, Valve2 optional;
  Sink         Sink1, Sink2;
connections
  Source1 -> Pump3 -> Pipe3 -> Pump2 -> Valve2 -> Sink2;
  Pump3 -> Pipe1 -> Pump1 -> Valve1 -> Sink1;
  Pump1 -> Pipe2 -> Valve2;
objective
  minimize costs (powerprice=0.24);
scenarios
  scenario daytime weight 50:
    Source1.out.pressure = 0;
    Sink1.in.flow        = 3;
    Sink1.in.pressure = 60;
    Sink2.in.flow        = 5;
    Sink2.in.pressure = 60;
end
```

Listing 3: Domain specific code for the example system.

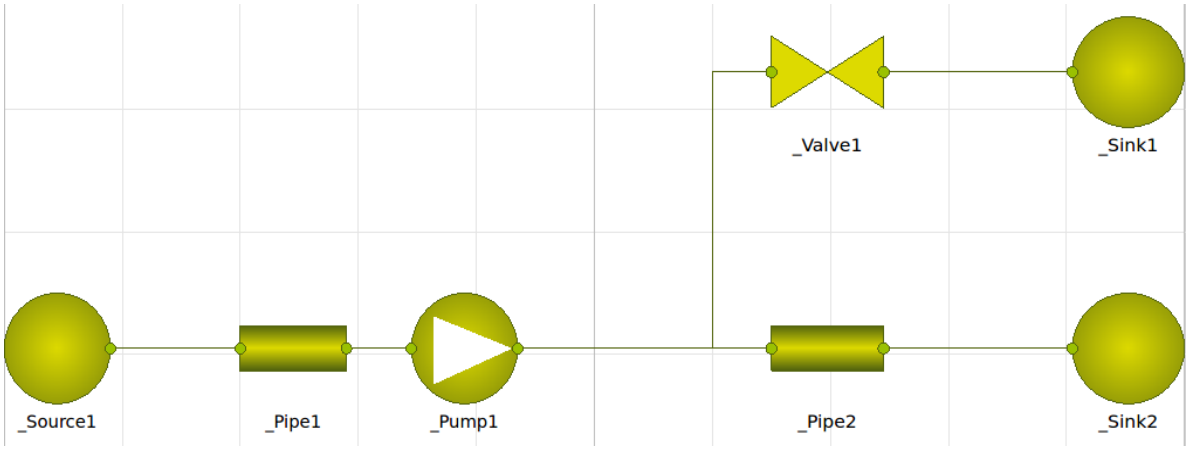
Experiments

The example system from Listing 2 was processed by our toolchain with two different scenarios. The settings and resulting layouts for those scenarios are given in Figure 5. The result suggests to add an additional sequential pump when the required pressure doubles, because one pump is not sufficient.

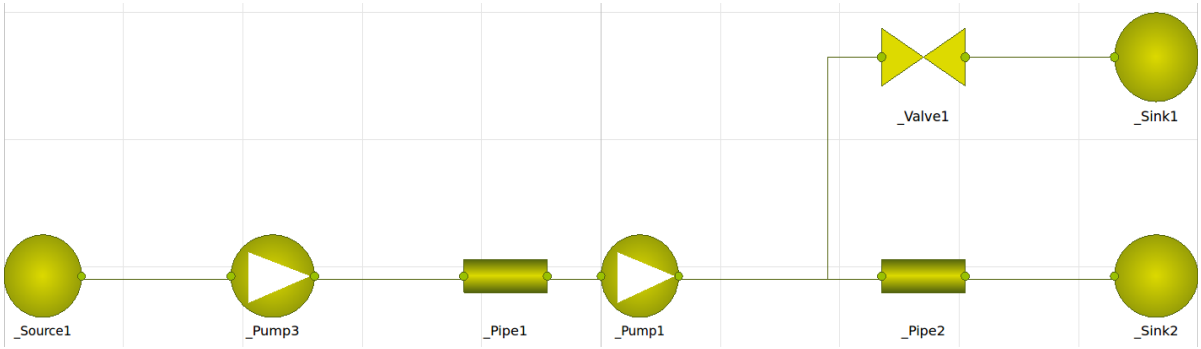
The layout pictures were taken from the simulation model and are computed only by information of Listing 2 and the solution of the optimization problem. Therefore, the arrangement of the sinks is reversed. Rotational speeds from the pumps are also computed but are not shown for simplicity.

The computing time for this example is between 10 and 20 seconds. The solver used was the free available software *SCIP* [6]. Test cases with more pumps and scenarios were not considered in this paper but are currently evaluated.

	Requirements Sink1		Requirements Sink2	
	flow	pressure	flow	Pressure
Setting A	3	60	5	60
Setting B	3	120	5	120



Optimal configuration for Setting A



Optimal configuration for Setting B

Figure 5: Results for two different settings A and B.

Conclusions

This paper presents a domain specific language for the design of optimal pump systems. Usually, energy consumption is minimized. The compiler generates a mixed-integer linear problem which is then solved by a free available MILP-solver. Additionally, a simulation model is generated and may be used to validate the results.

The language SHEP supports the specification of pump systems as well as of components used in such a system using domain specific terms. Pump characteristics for example can easily be described as a set of measuring points. The MILP formulation is automatically generated.

Connections, ports and components are validated. Furthermore, different load profiles can be specified. The compiler checks these load profiles such that the solver may not generate a trivial solution (e.g. no components and source providing the required pressure and flow). Instead of relying on a purely mathematical model the language SHEP allows to check domain-specific criteria.

The SHEP model annotated with a solution may also be reused as an architectural description. Varying variables from this solution may lead to different pump systems in another optimization process and could give more information about optimal pumping systems.

In future work the pumping system is analyzed in terms of possible values for flow and pressure. This information gives us a possibility to decide whether the generated MILP is feasible or infeasible. Furthermore, if it is known how much flow a pump must support, the corresponding characteristic can be tightened. This allows a reduction of the number of integer variables and can therefore decrease the solving time.

About 500 test cases were developed and systematically generated to research solving times and efficiency of compiler technologies.

Acknowledgement

This project is supported by the Federal Ministry for Economic Affairs and Energy with project sign 03ET1134C.

References

[1] P. Pelz, U. Lorenz, E. Thorsten, L. S. und G. Ludwig, „Designing Pump Systems by Discrete Mathematical Topology Optimization: The Artificial Fluid Systems Designer," International Rotating Equipment Conference.

[2] P. Pelz, *250 Jahre Energienutzung: Algorithmen übernehmen Synthese, Planung und Betrieb von Energiesystemen,* 2014.

[3] P. Pöttgen, T. Ederer, L. Altherr, U. Lorenz und P. Pelz, „Examination and Optimization of a Heating Circuit for Energy-Efficient Buildings," *Energy Technology,* 2015.

[4] U. Kastens, P. Pfahler und M. Jung, „The eli system," *Compiler Construction,* 1998.

[5] P. V. Juan, S. Ahmed und G. Nemhauser, „Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions," *Operations research,* Nr. 58, 2010.

[6] T. Achterberg, „SCIP: solving constraint integer programs," *Mathematical Programming Computation,* 2009.