Rule based decision support for the process flow

(SiMONE-Congress – 2008)

Dipl.-Ing. Dirk Lieser – WINGAS TRANSPORT GmbH & Co. KG
B. Comp. Sc. Mike Störmer – University of Paderborn

Abstract

On behalf of WINGAS TRANSPORT GmbH & Co. KG our workgroup had the challenge to develop a concept and to implement a tool for supporting the dispatchers with their work on minimizing the fuel gas consumption of compressor stations. Within this paper we try to formulate an approach to handle the complexity of the problem that results on closer examination of the process flow.

1. Introduction

Background. Increasing energy costs lead to a rising cost pressure for companies in transportation logistics. Therefore investment, respectively development work in software based applications for the optimization of transportation costs become more and more profitable. In the matter of pipeline-bound gas transport the modules developed by LIWACOM Informationstechnik GmbH provide initial appendage.

The Problem. The main defiance to the user is to serve the optimization module with the multiplicity of boundary conditions, as well as to bring his know-how about the complex coherences and interdependencies of network control into the workflow. In this context the users objective is to reduce the optimizers solution space with the objective of runtime optimization on the one hand and to produce a manageable and user-friendly solution set on the other hand.

Our contribution. To support the dispatcher in the dispatching center with his task, a supporting software application that helps to handle and to manage SiMONE optimization has been developed in a prototyping environment. Thereby the fundamental external and internal software qualities have been considered. User guidance and functional range has been developed according to wishes of the prospective users. It has been assumed that all users have extensive knowledge and experiences in pipeline-bound gas transport. Also the application is applicable in different planning processes and can work with different databases.
The primary objective of this prototyping was to develop and to validate a concept for the following realization inside of the control system of WINGAS TRANSPORT GmbH & Co. KG.

Organization of the paper. Within the section *problem area* we first introduce a few terms and definitions that a are needed for better understanding. Based on this terms the next subsection starts formulating the state of the art and results in the problem that is handled in this paper. The section *method of resolution* describes our contribution and ideas for solving the problem. For validating our solution a short prospect is given in the section *implementation*. Finally a summery und future prospects with suggestions for future tasks will complete our paper in the last section.

2. Problem area

2.1. Compressor terminology

First of all we need to describe the different terminologies that are in context with the term compressor. Starting on from the lowest level there we have the compressor and the driver. Compressor and driver are described by their characteristic maps and are compressing the gas as a compressor unit. On the next level we have the compressor plant. The compressor plant may contain one or more units. But at least one unit is required for definition. The highest level is the compressor plant which contains all compressor plants at the location. On this level pipes and valves are needed to ensure the connections. Also if there exist more than one compressor plant it is allowed that some of them may share a unit. Consider that in this case a unit can be used in one compressor plant at the same time only.

2.2. Compressor plant description

To understand our model of the compressor plants we have to introduce a few terms. In German they are called "Fahrweg", "Fahrweise" and "Astknoten". Representative translations are "piping setup", "operation mode" and "branch node".
A piping setup is based on the hydraulically topology and describes one special circuitry inside of a compressor plant.
The operation mode is based on an abstraction of the real hydraulically topology with valves as logical switches. It describes the type of interconnection of all inputs and outputs of an compressor plant with the help of setpoint charged control entities. These control entities are able to operate one ore more regulation elements depending on their setpoints.
To understand the description of the branch nodes we should have a look at figure 1 which shows a simple model of a compressor plant.

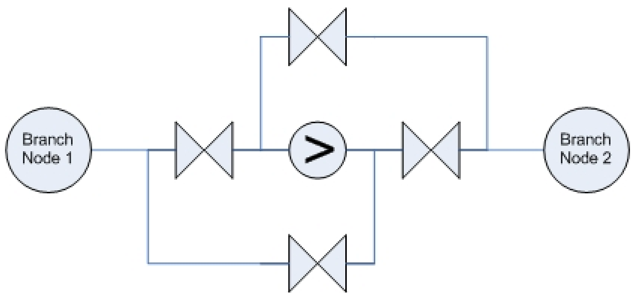

Figure 1: A simple compressor plant.

The compressor plant shown in figure 1 contains one compressor station, four valves and two so called branch nodes. The most important thing at this point is the relation between a compressor station and the branch nodes. All incoming and all outgoing connections of a compressor plant are done by branch nodes. Also every connected pipeline must have an own explicit branch node within the compressor plant. Additionally a branch node and a compressor station have to be separated by one unique valve. In the example shown this may result in two possible paths through the plant. One path is: *pipeline A* $\Rightarrow$ *branch node 1* $\Rightarrow$ *valve* $\Rightarrow$ *compressor station* $\Rightarrow$ *valve* $\Rightarrow$ *branch node 2* $\Rightarrow$ *pipeline B*. With raising complexity of a compressor plant we can show that the number possible paths will raise exponentially.

2.3. Network description

There exist three different types of pipeline networks that are in need to be differentiated. Taking its cue on the mathematical graph theory all edges of the networks are not necessarily directed. As we will show later in the paper the direction of an edge is only needed for analyzing and evaluating the flows.
The first type of a network is a simple pipeline without any crossings or circles. So every plant at maximum may have an input an output. Here a vertices (compressor plants) degree has a maximum of two for all inner vertices and a maximum of one for all boundary vertices (figure 2).

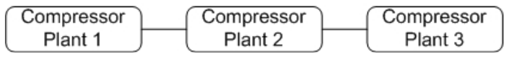

Figure 2: Compressor plants without crossings and circles (inline).

The second type is covered by a tree. In this case all vertices may have a degree grater then two. That means that the network may contain crossings. But an important constraint is the fact that no circles are allowed so far (figure 3). This one is the type of network that actually is covered by our contribution and therefore is handled in this paper. Nevertheless the second type can be decreased to the first one and therefore the first type is covered too.

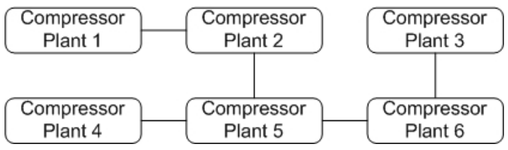

Figure 3: Compressor plants with crossings and without circles (tree).

The third and last possible type of network has no constraints (figure 4). Crossings and circles are allowed in a topology. In addition to figure 4 all compressor plants may have multiple connections to all other ones. For example an additional connection from plant two to plant six will also be possible.

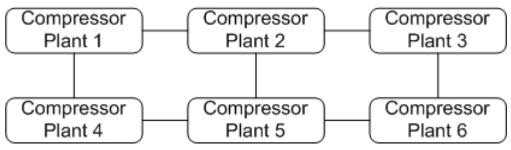

Figure 4: Compressor plant with crossings and circles (mesh).

Currently a meshed network is not tested yet. But a few theoretical analysis without considering the SiMONE solver allowed us to expect that enhancements around the implemented flow analysis will be needed to cover this model. Future requirements will show.

2.4. State of the art and the resulting problem

At the beginning we had to supply the SiMONE solver via the embedded graphical user interface. The first problem arose by feeding out all needed parameters for all relevant elements. In average we can say that every element is in need of about three to eight parameters. Especially the experimental changes of topological parameters are raising the work supremely. For about only two hundred elements nearly one thousand or more parameters have to be entered. So from now on the API comes into use. A configuration via API allows a comfortable configuration of the parameters. But with configuring the paths within the compressor plants and their interconnections the main problem of manageability arises. An inline compressor plant as shown in figure one will not be the problem but a compressor plant with six or more control entities and about five or more branch nodes will assign a nearly unsolvable task to the dispatcher. He has to find all feasible solutions and then has to figure out the reasonable ones. The following formula may help understanding the basic complexity.
Based on the objective model shown in figure 1 the number of branch nodes is called *B* and the number of compressor stations in a plant is called *CS*. Hence we get

$$\textbf{(1)} \qquad 2^{2*|B|*|CS|}, \forall \mathrm{B, CS} \mid \mathrm{B, CS} \; \varepsilon \; \aleph \wedge \mathrm{B} \geq 2 \wedge \mathrm{CS} \geq 1.$$

The only thing we do is to handle all valves as simple binary switches and to combine all possible position with each other. Thereby the number of switches is given by the number of

branch nodes *B*. Additionally we need to permute them over all included compressor stations *CS*. At this point our contribution starts helping the user to reduce the solutions. In a first step we automatically can reduce the number of all feasible solutions to reasonable solutions by formulating another formula.
Here we include the first limitation that only complete paths through a compressor plant are to be considered. That means that one or more incoming switches always are in need of getting one ore more unique outgoing switches to pipelines that are not used by incoming switches. Starting at this point means to apply the focus on a differentiation of incoming and outgoing switches.
Considering the outgoing switches only we get all possible combinations without attention to the sequence by

$$\sum_{k=1}^{N}\binom{N}{k}, \forall\ \mathrm{N, k \mid N, k}\ \varepsilon\ \aleph \wedge \mathrm{N} \geq 2,$$

with *N* as number of branch nodes. To get all possible combinations of corresponding outgoing switches we also do not need to attend the sequence. But here we have to note that only the outgoing switches of pipelines not used by incoming switches are available. Therefore we have to stop at *N-k* and get

$$\sum_{k'=1}^{N-k}\binom{N-k}{k'}, \forall\ \mathrm{N, k, k' \mid N, k, k'}\ \varepsilon\ \aleph \wedge \mathrm{N} \geq 2.$$

Combining the formulas for incoming and outgoing switches we have to arrange all possible incomings with all possible outgoings. Thus we get

$$\sum_{k=1}^{N}\left(\binom{N}{k} * \sum_{k'=1}^{N-k}\binom{N-k}{k'}\right), \forall\ \mathrm{N, k, k' \mid N, k, k'}\ \varepsilon\ \aleph \wedge \mathrm{N} \geq 2.$$

Now we have a formula that results in all possible paths through a compressor plant with one compressor station. Finally to make this formula compatible to compressor plants with more than one compressor station we have to permute the single compressor stations with each other and therefore get

(2)
$$\left(\sum_{k=1}^{N}\left(\binom{N}{k} * \sum_{k'=1}^{N-k}\binom{N-k}{k'}\right)\right)^{|CS|}, \forall\ \mathrm{CS, N, k, k' \mid CS, N, k, k'}\ \varepsilon\ \aleph \wedge \mathrm{CS} \geq 1 \wedge \mathrm{N} \geq 2.$$

Setting up a compressor plant with one compressor station and up to six branch nodes we get the following comparison:

B = N	***(1)***	***(2)***
2	16	2
4	256	50
6	4096	602

Table 1: Comparison of resulting solutions.

Remark that in this case the value of *CS* is set to one. Comparing *(1)* and *(2)* for *B=N=2* we get a manageable and correct amount of two options in case *(2)*. But case *(1)* let us recognize that also an inline compressor plant with only one compressor station and only tow branch nodes raises up the possible results extremely without any optimization.
As we can see still both results are not satisfying because the next problem is that normally a pipeline network contains more then only one compressor plant. And therefore we have to permute all possible solutions of a compressor plant with all possible solutions of each other compressor plant. With such a great number of resulting scenarios the solver will not finish its work in an arguable time.
What we got to do was to develop a concept and a corresponding process flow that logically reduces all possible combinations of valves as far as possible but additionally reduces the solvers solution space with use of the dispatchers know how. So we can formulate the thesis that we need a logically rule based system that can compute user made rules to shrink the first computed paths of a compressor plant by the users rules and finally supports the user coming to his decisions. In short a ruled based decision support.

3. Method of resolution

Within the second section we have learned some fundamentals and got an survey of the problem area. We showed that an implemented system can logically reduce the feasible paths to reasonable paths. Additional to section two we have to say that in reality nearly no compressor plant contains connections of every branch by every compressor station to every other branches. For example in reality the number of possible paths through a compressor plant with three branches and two compressor stations can be limited to around twenty. But permuting this results over five compressor plants will result in $20^5=3.2*10^6$ scenarios. Computing the best out of these scenarios still will blast the available time. So we have to introduce another step where an interaction between the user and the system can reduce these combinations as far as possible by the users know how.
At this point we defined a topology analyzing algorithm that builds up a knowledge base containing all reasonable paths for every compressor plant. Derived from a plant model similar to figure 1 this algorithm really is straight forward to implement based on the fundamentals of computer science. After the knowledge base is build up the user has to define rules for every plant that will allow selected paths being included in a scenario. The constraints for this rules depend on the flow rates at each branch node. To get the flow rates a preceding flow analysis has to be computed. The analysis of the flow rates here is the only point where the directions of edges are needed.
Due to the fact that from now on an interaction between user and system is taking place we defined the following process flow (figure 5). Thereby we consciously abstracted from surrounding processes that are not directly a part of the optimization. Following figure 5 the

processes are described in detail. Also the description will show how we continuously can minimize the solution space.

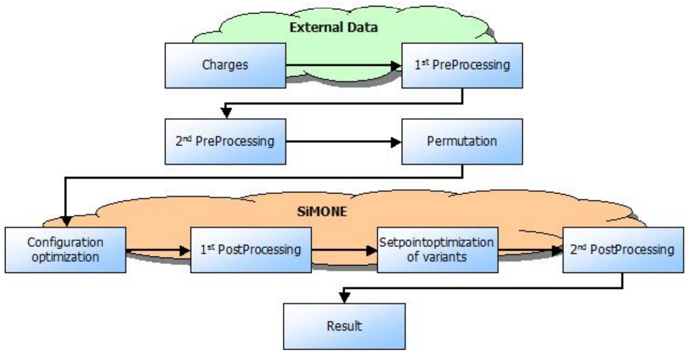

Figure 5: Process flow.

Charges. The charges are required quantities for every supply and exit nodes within a scenario. They are valid for all resulting variants of a scenario. There may exist several sources that can deliver the charges. The target of this process is to generate a list of all network elements that are in need of flow rates.

1st PreProcessing: This process can be split into three steps. The flow analysis has to create a balanced flow scenario and the flow rates for every compressor plant have to be assigned. The second step is for assigning the pressure rates to the nodes depending on the flow rates. Also the head pressure of the storage is assigned in this step. Here the user has to maintain the master dates with default pressures as well as a formula that is needed by the system to compute the pressure rate for the storage. For future use an interface in the last step is provided that will allow to supply the system with several values that directly are read out of the supervising system. For example this values are flow rates, operation modes, pressure rate or weather data. In this case the first two steps can be ignored because all values needed will come out of the supervising system. In summery the target of this process is to create a balanced load scenario that contains all relevant boundary conditions. For the maintenance of the data an additional interface is provided.

2nd PreProcessing: After the charges und boundary conditions have been assigned in the first preprocessing the user made rules have to be applied in the second preprocessing. This operation will reduce the amount of reasonable paths within a compressor plant again to a number of paths that fulfill the boundary conditions. To decrease the amount of paths per compressor plant again the user has the chance to chose several paths out of the computed ones in the last step. To test individual paths or if no rule is fulfilled the user can add paths out of the complete pool of reasonable paths. Currently we decided an upper limit of five combinations of paths. In this process we try to reach the lowest possible solution space to minimize the calculation time without missing a potentially profitable option.

Permutation: This process has the task to build up the permutations of all plant operation modes. We can do so because all compressor plants operate independent. In the worst case we will get an amount of $O(n^k)$ possible scenarios for computing with n as amount of operation modes per compressor plant and k as amount of compressor plants.
Five operation modes and ten compressor plants will result in 5^{10} scenarios. Therefore a reasonable limitation is indispensable.

ConfigurationOptimization: Continuing at this point the optimizer of SiMONE is started the first time in configuration optimization mode. Therefore all scenarios are transferred via the API. As result we get the configurations of the used compressor stations including the setpoints. The configuration of a compressor station is to be understood as the amount of default compressor units. That means that the user has to choose a default type compressor unit for every compressor station. The resulting value is the number of default compressor units needed in a compressor station. This value is computed in an interval from *0* to *n* depending on the maximum power of all real compressor units in a compressor station. The developed system automatically limits the value. A mixed integer and discrete optimization is done for the entire network. So the result is a first suggestion for the networks operation mode. Remark that the simulation only uses units of the same type for a compressor station.

1st PostProcessing: After the results of the configuration optimization are present they have to be prepared for the next step. Therefore the configurations computed by SiMONE must be substituted by configurations that are possible in reality. For example SiMONE computes that two units with 25MW each are needed to produce the required 35MW in the compressor station. In reality there only one unit with 25MW is available but additional two units with 10MW each are existing. Now the user has to choose the combination of the units. In this case the user can choose the 25MW unit together with a single 10MW or together with both 10MW units. As a result the user will get two new options that may fit an optimal state. Thus for every running compressor plant there will raise new variants that have to be permuted with each other variants of all compressor plants and then have to be computed by the simulation.
Here we decided that the maximum amount of variants per compressor plant is limited to two. This restriction is done to keep the solution space as small as possible.

SetpointOptimization of variants: The variants created during the first post processing again must be send to the simulation via API. Now the mode setpoint optimization is used. SiMONE itself has an operation mode that can do an setpoint optimization of variants. But first tests of this mode showed that managing and controlling of the variants is not effective enough for our requirements. Therefore we decided to use the setpoint optimization without the variants objective and to manage the variants outside of SiMONE.

2nd PostProcessing: Similar to the first post processing the results and initial conditions of a number of best scenarios are read out in this process. Currently this is realized by a routine that uses the SMONE API.

Result: In this process the system submits the results of the second post processing to the user. Here the user has the chance to compare the resulting scenarios with each other. There exist several tools for analysis. After this process further processing depends on the users task.

To demonstrate the reduction of the solvers solution space the following table shows an example that may arise in reality. Therefore we define a network that consists of two compressor plants containing two compressor stations and four branch nodes each.

process	*solutions*
initial	$4.3*10^9$
optimized	$6.3*10^6$
flow analysis & rules	25
2nd preprocessing	4

Table 2: Reduction during the process flow.

4. Implementation

For testing and validating our concept with real conditions we implemented a prototype of a program. Due to the papers focus we only would say a few words concerning the implementation. Further questions around the implementation can be asked in the congresses environment.
The prototyped implantation is done in VBA within the Microsoft Excel environment. This decision has been done because of the simple form for connecting the API of SiMONE and although because many needed functions are still included in the provided environment. The program itself is done by a module based architecture. Hence all single processes can be handled as modules and therefore different modules and ideas easily can be tested or interchanged. Finally the following figure will show a short assembly of the architecture (figure 6).

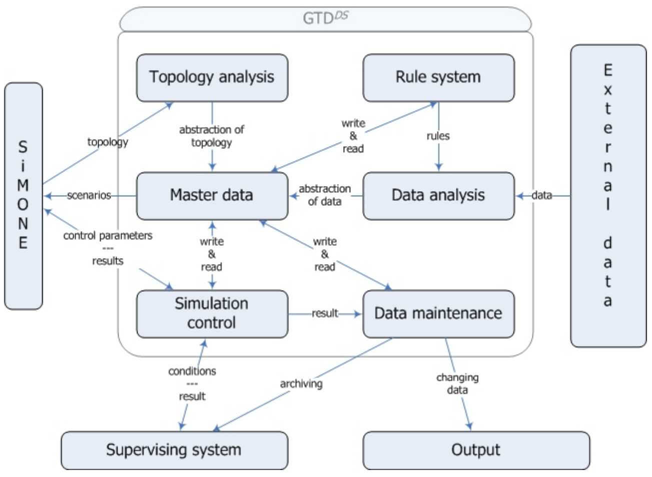

Figure 6: Assembly of the prototyping system.

5. Summery and future prospects

Coming to an end we can say that in the beginning the solution space was exorbitant. But there exist many options where the solution space can be reduced. The combination of logical processes implemented within a control program and the users know how can shrink the options that span the solution space to a value in the range of lower double figures.

For future use we have learned during prototyping that needful work can be done in the direction of auto-adaptive rules. Because of the complexity of a pipeline network the user really has a hard job with configuring all rules. If the conditions of rules are to strong reasonable options can be excluded. Otherwise if the conditions are to casual the resulting solution space can be to wide. Maybe transferring the rules into fuzzy rules will help too.
Another conclusion concerns the user interface of the implemented tool. Currently the user interface is similar to a text based dialog system. Therefore it is not easy to follow the dialog without any graphical feedback. An upcoming idea is to implement an abstract graphical model for possible sub-paths that can be compared to the preview pans used in route guidance systems to show the next action.