Entwurf und Implementierung einer Sprache zur Planung optimaler Pumpensysteme

Benjamin Saul, Wolf Zimmermann

Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1
06120 Halle (Saale)

Zusammenfassung. In diesem Artikel wird eine Sprache und deren Umsetzung für die Planung optimaler Pumpensysteme zur Förderung von Flüssigkeiten vorgestellt. Die entwickelte domänenspezifische Sprache beschreibt Komponenten wie z. B. Pumpen und die zugehörigen Anforderungen. Wir beschreiben wichtige Sprachkonzepte und gehen insbesondere auf die Generierung eines linearen Problems zur Optimierung ein. Die Optimierungsaufgabe eines solchen Systems liegt darin, die bestmögliche Auswahl an zu installierenden Pumpen zu treffen, um möglichst energie- oder kosteneffizient die Anforderungen zu erfüllen. Vor und während der Generierung dieser Optimierungsaufgabe müssen Analysen ähnlich zu Programmanalysen erfolgen, um möglichst detaillierte Fehler- und Diagnosemeldungen zu generieren.

1 Motivation

In vielen industriellen Fertigungs- und Produktionsanlagen sowie Versorgungsnetzwerken kommen Pumpensysteme zum Einsatz. Ein Pumpensystem ist eine Anlage mit Pumpen, Rohren und diversen Armaturen, deren Zweck es ist, Flüssigkeiten mit Anforderungen an Druck und Durchfluss über eine Distanz zu transportieren. Eine Beispielanlange ist in Abbildung 1 dargestellt. Es gibt fest vorgegebene Komponenten, wie Behälter und Abnahmestellen, mit spezifischen Drücken und Durchflüssen, die entweder durch die Umgebung vorliegen oder erst erreicht werden müssen. Das Erreichen dieser Drücke geschieht durch den Einbau und Betrieb diverser Pumpen.

Das Betreiben von Pumpen stellt zusammen mit sonstigen Fördermethoden wie Fließbändern 37% der in Deutschland benötigten Energie dar [12]. Daher sollen die Anlagen möglichst so geplant werden, dass der Stromverbrauch und die Anlagenkosten möglichst gering sind. Optimierungen von industriellen Anlagen können Einsparpotentiale von bis zu 90% der Pumpenenergiekosten erzielt werden [4]. Um dieses Potential voll auszunutzen, wurde in [11] ein Verfahren entwickelt, mit dem man optimale Pumpensysteme berechnen lassen kann. Dabei muss ein abstraktes Pumpensystem, also sämtliche Anordnungen und Konfigurationen von Pumpen und Armaturen, als mathematischen Optimierungsproblem

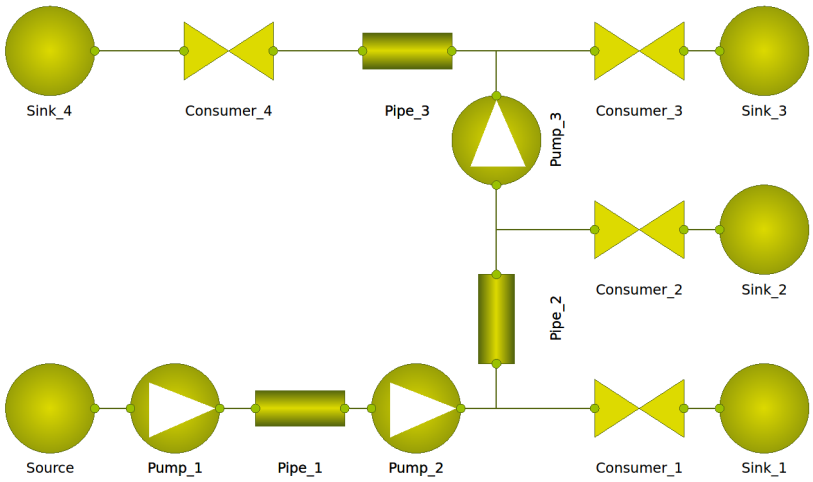

Abb. 1. Beispiel eines Pumpensystemes mit verschiedenen Pumpen und Abnehmern. Aufgabe ist es, einen Strom von der Quelle zu den Senken zu erzeugen und dabei bestimmte Drücke bei den Abnehmern zu erzielen.

formuliert werden. Dieses besteht zu einem großen Teil aus verschiedenen Variablen und Gleichungen, ist also für einen Pumpenanlagenplaner nur schwer handhabbar. Diese müssen dort Variablen für einzelne Pumpen definieren und aus dem Ergebnis der Optimierung herauslesen, wie die optimale Anlage beschaffen sein muss.

Um den Architekt der Pumpenanlage zu unterstützen und das Entwerfen einer optimalen Anlage zu einem durchgängigen Prozess zusammenzufassen, entwickeln wir die domänenspezifische Sprache SHEP (Sprache für hocheffiziente Pumpensysteme). Den zugehörigen Übersetzer entwickeln wir mit der Werkzeugsammlung Eli [8]. In der Sprache können Komponenten des Pumpensystems, wie Rohre, Pumpen und Ventile, als Typen beschrieben werden. Die darin geltenden physikalischen Gesetzmäßigkeiten sind bereits vordefiniert. Für das System können nun einzubauende Komponenten bestimmt werden. Mit Typprüfungen wird dann unter anderem bestimmt, welche Komponenten des Pumpensystems miteinander verschaltet werden können. Die aktuelle Hauptaufgabe des Übersetzers ist es, ein Optimierungsmodell aus der domänenspezifischen Beschreibung zu erstellen. Dieses wählt dann unter den möglichen Pumpen diejenigen aus, welche im optimalen System eingesetzt werden sollen. Die benötigten Aspekte des Übersetzers werden in Abbildung 2 dargestellt.

Erzeugen von linearen Problemen aus domänenspezifischen Sprachen Hierbei kommt es im Wesentlichen darauf an, dass Verhaltensgleichungen möglichst günstig beschrieben werden, damit die Optimierung schnell durchgeführt werden kann. Die Attribute der einzelnen Komponenten müssen als Variable und Parameter des Optimierungsproblems formuliert werden. Ent-

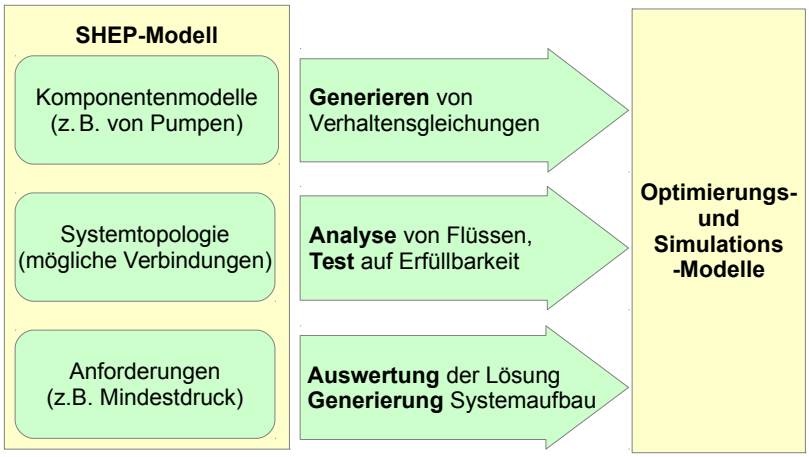

Abb. 2. Übersicht über die Werkzeugkette zur Erstellung optimaler Pumpensysteme.

sprechende Gleichungen für z. B. Flusserhaltung müssen generiert und linearisiert werden.

Analyse des Flüssigkeitsstromes und Lösbarkeitstest Durch Vorabanalysen im System werden unzulässige Verbindungen detektiert, welche die daran anliegenden Drücke oder Durchflüsse nicht unterstützen. Der Test auf Lösbarkeit ist notwendig, denn die Fehlermeldungen eines Lösers sind für den Systemarchitekten unzureichend. Die Fehlermeldungen geben nur die Unlösbarkeit an, ohne deutlich machen zu können, welche Druckanforderung den Widerspruch ausgelöst hat. Das Wissen der Domäne kann hier detaillierte Fehlermeldungen möglich machen.

Generieren des optimierten Pumpensystems aus der Lösung Ergebnis des Lösers sind die Belegungen der Variablen, insbesondere der Entscheidungsvariablen. Diese zu interpretieren und auszuwerten ist manuell schwierig und fehleranfällig. Daher müssen diese im Übersetzer gesammelt werden und daraus das endgültige optimierte geplante Pumpensystem formuliert werden. Dieses wird als Simulationsmodell beschrieben, um überprüfende Berechnungen darauf durchführen zu können.

Im nächsten Abschnitt dieses Artikels werden andere Arbeiten diskutiert, die sich mit der Generierung von Optimierungsproblemen beschäftigen. Außerdem wird dabei die hier eingesetzte Werkzeugsammlung Eli vorgestellt. Im darauf folgenden Abschnitt werden ausgewählte Sprachbestandteile und die dabei auftretenden Typprüfungen vorgestellt. Abschließend wird auf die Übersetzung in ein Optimierungsproblem bzw. Simulationsmodell eingegangen.

2 Verwandte Arbeiten

Die Entwicklung domänenspezifischer Sprachen wird durch diverse Werkzeuge und Programmierrichtlinien unterstützt [7]. In diesem Projekt wird mit der

Werkzeugsammlung Eli gearbeitet, um einen Übersetzer von der Systembeschreibung zum Optimierungs- bzw. Simulationsmodell zu generieren [8]. Eli verfügt über die Möglichkeiten, leicht zu erzeugende Fehlermeldungen ausgeben zu können [3]. So kann beispielsweise angezeigt werden, an welchen Stellen des Pumpensystems ein geforderter Höchstdruck konstruktionsbedingt überschritten wird. Dadurch ist es möglich, gezielt Fehlermeldungen bezüglich der Nichtlösbarkeit des Optimierungsproblems zu erzeugen. Der Anwender sieht so direkt, welche seiner geforderten Systembedingungen oder Verschaltungen zu Widersprüchen führen.

Die grundlegende Formulierung der Optimierungsaufgabe und die Vorstellung der Anwendungsdomäne erfolgt durch unsere Projektpartner in [11]. Dort erfolgt die Modellierung des Problems allerdings noch direkt mit mathematischen Modellierungssprachen wie AMPL [5]. Diese unterstützt zwar schon eine Zusammenfassung von Gleichungen durch Iteration über Mengen, allerdings wird das Optimierungsproblem immer noch mathematisch formuliert und es erfordert ein hohes Verständnis von der Modellierung, falls man diese Umsetzen möchte. Das ausgegebene Ergebnis kann durch die Sprache beeinflusst werden. Somit können hier auch komplexe Ausgaben basierend auf der Lösung angezeigt werden. Hilfreich bei der Modellierung in AMPL ist, dass eine Trennung von dem beschriebenen System mitsamt der geltenden Gleichungen und den tatsächlichen Daten erfolgt.

Dadurch ist es möglich, dass die notwendigen Gleichungen und Ausgabeanweisungen einmalig erstellt werden und anschließend diverse Szenarien, also Lastprofile, umgesetzt werden können. Beim Eingeben der verwendeten Komponenten und Daten müssen die Gleichungen dann nicht nochmal verändert werden. Allerdings ist die Möglichkeit der semantischen Prüfung stark eingeschränkt und Vorabprüfungen auf Lösbarkeit sind quasi nicht vorgesehen. Dies ist dadurch bedingt, dass LP-Löser eine allgemeine Software sind, welche jedes lineare Problem bearbeiten müssen. Lineare Probleme sind eine Menge von uninterpretierten Gleichungen und Ungleichungen, deren Bezug zum abgebildeten Modell erst im Nachhinein erzeugt wird. Betrachtet man nur dieses, so lassen sich schwer Rückschlüsse auf das Pumpensystem herstellen.

Strategien zur Generierung von linearen und nichtlinearen Optimierungsproblemen wurden in [10] untersucht. Hierbei liegt die Formulierung des Ausgangsproblems allerdings noch nah am linearen Programm. Allerdings können damit schon sinnvolle Unterstrukturen gebildet werden, welche die Modellierung vereinfachen und die Modelle lesbarer machen. Aktuellere Vorgehen beinhalten eine Modelltransformation zur Erzeugung von linearen Problemen von abstrakteren Modellen ausgehend [6]. Dabei wurde die selbstständige Laufzeitoptimierung von Programmen untersucht. Aus einer domänenspezifischen Sprache, die die einzelnen Implementierungen beschreibt, wird ein Optimierungsproblem erzeugt. Auch hier erfolgt eine kombinatorische Explosion des Lösungsraumes bei der Verwendung mehrerer Komponenten, was sich in einem Anstieg der benötigten Rechenzeit zum Finden einer Lösung niederschlägt. Eine semantische Vorab-

prüfung auf die Lösbarkeit des erstellten Optimierungsproblems erfolgt nicht. Die Lösung muss manuell weiterverarbeitet werden.

Eine andere Vorgehensweise der Optimierung von Pumpensystemen kann man z. B. in [1] finden. Hier wird die Simulationssprache modelica derart erweitert, dass der Löser für die Gleichungssysteme der modelica-Umgebung nicht nur das Verhalten simuliert, sondern außerdem entsprechende Parameter optimiert. Der Nachteil hierbei ist, dass nur Parameter optimiert werden, nicht ganze Strukturen von Verschaltungen wie in unserem Projekt. Es muss also direkt bekannt sein, welche Pumpen in welcher Verschaltung verwendet werden. Außerdem ist wieder ein recht hohes Verständnis der Sprache modelica sowie ihrer Erweiterung erforderlich. Auf die Domäne der Pumpensysteme angepasste Fehlermeldungen sind ebenfalls nur schwer möglich, weil modelica für die Modellierung beliebiger hybrider Systeme eingesetzt werden kann. Eine Auswertung der Lösung entfällt, da die berechneten Parameter direkt ein Simulationsmodell bzw. das fertige System ergeben.

Basierend auf den bisher gemachten Erfahrungen wird für das Pumpensystem zunächst ein GMPL-Modell des zugehörigen linearen Optimierungsproblems erstellt. GMPL ist eine Teilsprache von AMPL, welche für unsere Zwecke ausreichend und frei verfügbar ist. Entsprechende Solver findet man z. B. unter [9]. Wir verwenden die Übersetzerbauwerkzeugsammlung Eli, um einen Übersetzer zu entwickeln, der aus unserer domänenspezifischen Sprache ein Optimierungs- und schließlich Simulationsmodell mit den energieoptimalen Pumpen erstellt.

3 Besondere Sprachbestandteile und deren Übersetzung

Im Folgenden werden einige Sprachkonzepte herausgegriffen und näher betrachtet. Dabei wird das Konzept als solches vorgestellt und auf Details bei der Generierung der zugehörigen Optimierungs- und Simulationsmodelle eingegangen.

3.1 Spezifikation von Pumpentypen

Eine Pumpe, hier speziell die Kreiselpumpe, ist eine hydraulische Komponente zum Druckaufbau, welche in Abbildung 3 schematisch dargestellt wird. Sie besteht aus einem Elektromotor, der die über einen Frequenzumrichter aufgenommene elektrische Energie in mechanische Energie als Drehbewegung von Rotorblättern umwandelt. Diese Rotorblätter drücken die Flüssigkeit in Förderrichtung durch die Leitungen, wodurch diese dort unter Druck gesetzt wird. Der eingehende Volumenstrom in $\frac{m^3}{h}$ bleibt erhalten, beeinflusst aber die benötigte Energie, um Druck aufzubauen. Durch eine Regelung der eingehenden elektrischen Leistung über den Frequenzumrichter kann die Rotationsgeschwindigkeit der Rotorblätter eingestellt werden. Die Pumpe erzeugt dann bei kleinerer Stromzufuhr auch weniger Druck.

Die entwickelte domänenspezifische Sprache SHEP ist objektorientiert. Die Typen der Sprache sind Pumpen, Rohre und Armaturen, welche definiert werden können. Ein Beispiel für eine Pumpenspezifikation vom Typ `PumpTypeA` ist im

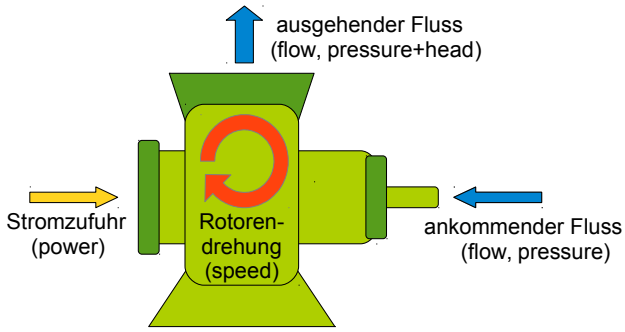

Abb. 3. Schematische Darstellung einer Kreiselpumpe. Durch Zufuhr von elektrischer Energie am Frequenzumrichter bzw. Motor wird eine Rotation ausgelöst, die einen Druck am Ausgang erzeugt und die Flüssigkeit fördert. Durch Regelung der Leistungsaufnahme werden Drehzahl und somit Druckaufbau eingestellt.

Codeausschnitt 1 zu sehen. Zur Spezifikation eines neuen Pumpentyps gehört die Angabe der Kennlinien, eventuelle Kosten der Pumpe sowie die Spezifikation der vorhandenen Anschlüsse der Pumpe.

```
pump PumpTypeA
  characteristic
    speed     head      flow     power;
     400       1.1      0.00      5.00;
     400       1.05     0.78     10.00;
     400       0.985    1.55     15.00;
     400       0.83     2.33     20.00;
     400       0.60     3.10     25.00;
     700       2.94     0.00     20.00;
     700       2.84     1.28     25.00;
     700       2.645    2.55     35.00;
     700       2.24     3.83     50.00;
     700       1.63     5.10     55.00;
   // ... ... ... ...
  costs
    purchase = 100;
  ports
    in  Flange(DN 32, PN 10);
    out Flange(DN 32, PN 10);
end
```

Codeausschnitt 1. Spezifikation eines Pumpentyps mit dem Namen PumpTypeA, angegebenen Kennlinien, Kosten und den vorhandenen Anschlüssen

Die Kennlinien werden in SHEP als Menge von Messpunkten angegeben. Sie stellen das Verhältnis zwischen Durchfluss, Druckerhöhung, Stromverbrauch

und Drehzahl der Rotorblätter dar. Grafisch dargestellt werden diese jeweils in einzelnen Diagrammen, bei denen Kennlinien einer festen Drehzahl gezeichnet werden, die dann den Durchfluss mit der Messgröße vergleichen. Dies wird in Abbildung 4 dargestellt. Bei derartigen Messungen wird die Pumpe bei festen Drehzahlen unterschiedlich großen Volumenströmen ausgesetzt. Dabei misst man den aufgebauten Druck und den Stromverbrauch. Als Einheiten verwenden wir Standardeinheiten.

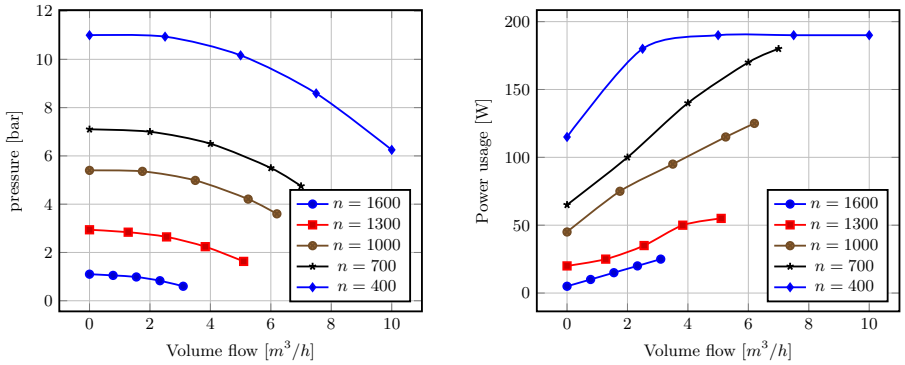

Abb. 4. Beispiel von Pumpenkennlinien in Relation zu einer festen Drehzahl n in Umdrehungen pro Minute.

Bei den Kosten wird der Einkaufspreis bzw. die Installationskosten angegeben. Außerdem können hier andere zeitabhängige laufende Kosten, z. B. Wartung und Stromverbrauch, mitsamt Gewichtung angegeben werden. Diese bilden aufsummiert die ökonomischen Kosten für das System.

Anschlüsse werden mit den Schlüsselwörtern `in` und `out` gekennzeichnet. Der Typ `Flange` steht für die davor definierte Anschlussart Flansch. Dieser hat einen Durchmesser (DN) und einen zulässigen Höchstdruck (PN). Damit kann man die Kompatibilität zu anderen Anschlüssen und zum Gesamtsystem überprüfen.

In Abbildung 5 wird die innerhalb des Übersetzers definierte Klassenhierarchie dargestellt. Jede Klasse verfügt über die dafür typischen Attribute und Gleichungen. Innerhalb der Hierarchie können neue Typen definiert werden. Beim Erben werden dabei die Eigenschaften der übergeordneten Typen bzw. Klassen übernommen. Anschlüsse der Komponente können weiter konkretisiert werden. Die Klassenhierarchie wird dann in ein Teilmengenkonzept der Sprache GMPL überführt. Die entsprechenden Klassen bleiben in ihrer Struktur erhalten.

Der Aufbau des GMPL-Modelles der oben beschriebenen Pumpe wird in Codeausschnitt 2 gezeigt. Den übernommenen Bezeichnern wird ein Unterstrich vorangestellt, um etwaigen Identitäten mit vorhandenen Bezeichnern zu entgehen. Der Unterstrich ist folglich am Anfang eines Bezeichners in SHEP unzulässig. Mit den `set`-Anweisungen wird die Pumpe in die Komponentenhierarchie eingeordnet. Anschließend erfolgen Definitionen für die benötigten Varia-

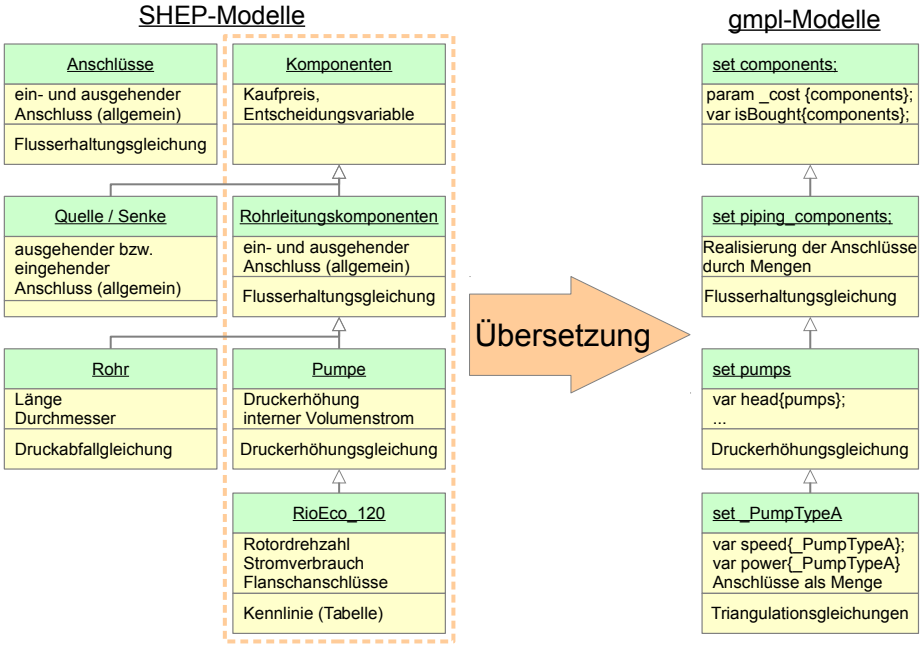

Abb. 5. Übersicht über Hierarchie der in SHEP definierten Komponenten und dem beschriebenen Pumpentyp PumpTypeA. Die dargestellten Einheiten repräsentieren die Klassen mit ihrem Namen, den darin definierten Variablen und den Gleichungen. Die Pfeile repräsentieren die Vererbungsbeziehung zwischen den Klassen.

blen, welche teils für alle Pumpen und teils nur für diesen Pumpentyp angelegt werden. Die Kennlinien werden als zweidimensionale Mengen modelliert, welche jeweils dem Tupel `(flow, head)` entsprechen. Dies ist aufgrund der Beschaffenheit der Kennlinien eindeutig, da bei gleichem Volumenstrom und gleicher Druckdifferenz auch der gleiche Energieverbrauch bzw. Rotorendrehzahl auftreten muss. Diese Werte werden dann den Tupeln zugeordnet. Mit den definierten Variablen und Parametern können schließlich die Gleichungen definiert werden, die für alle Pumpen gleichermaßen gelten.

Die Variablen der Kennlinien werden durch andere Gleichungen miteinander in Beziehung gesetzt. Um diese zu modellieren stehen verschiedene Verfahren mit unterschiedlicher Laufzeit beim anschließenden Lösen des Problems zur Verfügung. Die Wahl des Verfahrens und dessen Parameter werden beim Übersetzungsprozess angegeben. Da es unterschiedlich detaillierte Linearisierungen gibt, kann dies einen Einfluss auf das Ergebnis der Optimierung ausüben.

Bei der Generierung des Simulationsmodells in modelica können die in der Pumpe definierten Attribute als Variablen übernommen werden. Für den Druckaufbau jedoch muss eine Funktion aus den Kennlinien ermittelt werden, abhängig vom Volumenstrom und den zusätzlichen Einträgen. Diese bestimmt dann das simulierte Verhalten der Pumpe. Der Unterschied zum GMPL-Modell ist der,

```
# Definition der Menge mit deklarierten Pumpen
set pumps within piping_components;
set _PumpTypeA within pumps;

# Verwendete Variable
var _power { scenarios, _PumpTypeA };
var _speed { scenarios, _PumpTypeA };
var _head { scenarios, pumps };
var _flow { scenarios, pumps };
var _pressure_port { scenarios, pipe_connectors };
var _flow_port { scenarios, pipe_connectors };

# Definition der Kennlinienparameter
set    basicvalues_PumpTypeA dimen 2; # flow, head
param basicvalue_PumpTypeA_speed {basicvalues_PumpTypeA};
param basicvalue_PumpTypeA_power {basicvalues_PumpTypeA};

# Eingabe der Kennlinienwerte
param: basicvalues_PumpTypeA:
          basicvalue_PumpTypeA_speed
          basicvalue_PumpTypeA_power
    := 0.00    1.1      1300      5.00
       0.78    1.05     1300     10.00
# ... ... ... ...

# Gleichungen fuer den Druckaufbau
PressureIncrease {S in scenarios, P in pumps}:
  _pressure_port[S, P, '_out']
   = _pressure_port[S, P, '_in'] + _head[S, P];
```

Codeausschnitt 2. Auszug aus dem erstellten GMPL-Code für den Pumpentyp PumpTypeA mit den Werten der Kennlinie, auftretenden Variablen und Gleichungen. Durch die vordefinierte Menge pumps können Gleichungen vereinfacht werden.

dass man sich bei modelica nicht auf lineare Funktionen beschränken muss, was eventuell eine realitätsnähere Abbildung darstellt. Auch dieses geschieht über externe Funktionen zur Interpolation und auch hier gibt es dafür mehrere Varianten der Umsetzung, die gewählt werden sollten.

3.2 Spezifikation des Systems

Dem Spezifizieren des Systems entspricht das Zeichnen eines Verschaltungsplanes für die Pumpenanlage. Hier werden Komponenten eingebracht, benannt, mit Parametern versehen und verbunden. Schließlich werden Anwendungsfälle spezifiziert, die mit verschiedenen Variablenbelegungen daherkommen. Das Beispielsystem wird in Abbildung 6 gezeigt. Die zugehörige Systembeschreibung findet sich im Codeabschnitt 3.

Zunächst werden die Komponenten definiert. Dabei stehen `Source` und `Sink` für abstrakte Quellen und Senken, um von der konkreten Umgebung unabhängige

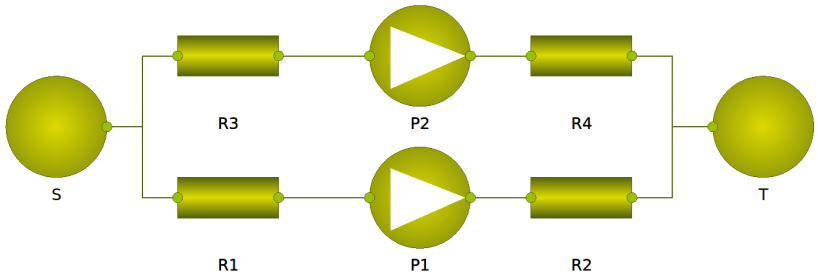

Abb. 6. Beispielsystem aus dem Codebeispiel 3.

```
system MySys
  Source       S;   // Verwendete Komponenten
  PumpTypeA P1;
  PumpTypeA P2 optional;
  SteelPipe  R1, R2, R3 optional, R4 optional;
  Sink         T;
connections          // Verbundene Komponenten
  S  -> R1 -> P1 -> R2 -> T;
  S  -> R3 -> P2 -> R4 -> T;
objective
  minimize costs; // Zielfunktion (hier: Summe aller Kosten)
scenarios      // Verschiedene Anwendungsfaelle mit Wichtung
  scenario S1 weight 3:
    S.out.pressure  = 3;
    S.out.flow      = 0.78;
    T.in.pressure   = 4.05;
  scenario S2 weight 1:
    S.out.pressure  = 4;
    S.out.flow      = 3.50;
    T.in.pressure   = 8.985;
end
```

Codeausschnitt 3. Konfiguration eines einfachen Systems, bei dem zwischen Quelle und Senke entweder eine oder zwei parallel platzierte Pumpen liegen. Ob beide Pumpen notwendig oder kostengünstiger sind, ergibt sich nach den angegebenen Lastprofilen in den Szenarios.

Teilsysteme zu beschreiben. Dazwischen befindet sich eine Menge von Pumpen und Rohren, von denen einige optional sind, also nicht im optimalen System vorhanden sein müssen. Anschließend werden die Komponenten durch den Pfeil-Operator `->` miteinander verbunden. Dabei wird jeweils der ausgehende Anschluss der linken mit dem eingehenden Anschluss der rechten Komponente verbunden. Diese Werte beeinflussen z. B. auch den möglichen Rohrinnendruck. Daher können sie bei fehlerhafter Zusammenschaltung das Optimierungsproblem unlösbar machen. Eine Analyse der Anschlüsse ermöglicht es, direkt an den auftretenden Stellen Widersprüche kenntlich zu machen.

Die Zielfunktion beschreibt die zu optimierenden Kriterien. In diesem Fall hier werden die Kosten insgesamt minimiert, es werden also die Variablen aus den `costs`-Abschnitten aller Komponenten aufsummiert und diese Summe minimiert. Auch andere Zielkriterien sind hier möglich, müssen dann aber manuell beschrieben werden.

Die Anwendungsfälle darunter beschreiben die notwendigen Anforderungen an das System. Unterschiedliche Szenarien können z. B. Tag- und Nachtbetrieb beschreiben, jeweils mit eigenen Ansprüchen an Druck und Durchfluss im System. Hier kann man durch eine semantische Prüfung fordern, dass die wichtigen Eingangs- und Ausgangsparameter der Quellen und Senken gegeben sind. Wird nämlich der Volumenstrom nicht spezifiziert, so nimmt die Optimierung möglicherweise unsinnige Werte dafür an.

Ein Auszug des erzeugten GMPL-Codes für das Beispielsystem findet sich in Codeausschnitt 4. Komponenten werden als Elemente der entsprechenden Typmengen definiert. Verbindungen können direkt als Tupel von zwei Komponenten und ihren Anschlüssen modelliert werden. Die Gleichungen für die Verbindungen sind für alle Modelle gleich, werden also generiert. Gleichungen der Szenarios können nahezu direkt übernommen werden, müssen nur syntaktisch angepasst werden. Für die Abbildung optionaler Komponenten wird im Optimierungsproblem jeder Komponente die Entscheidungsvariable `_isBought` zugewiesen. Da jede Komponente eine Entscheidungsvariable besitzt, müssen nichtoptionale Komponenten auch als solche gekennzeichnet werden. Die Entscheidungsvariable wird dann `1` gesetzt. Die Zielfunktion wird aus der Angabe in SHEP generiert.

Für alle optionalen Komponenten entscheidet die Optimierung, ob diese verwendet werden oder nicht. Dazu tritt die Entscheidungsvariable in den verschiedenen Gleichungen auf und macht diese ungültig, falls die Komponente nicht benötigt wird. Diese Methoden werden in [11] beschrieben. Bei der Generierung des Optimierungsproblems ist dabei zu beachten, dass solche Entscheidungsvariablen ganzzahlig sind. Dadurch nehmen sie Einfluss auf die Optimierungsgeschwindigkeit. Die Anzahl dieser Entscheidungsvariablen hat also einen wesentlichen Einfluss auf die Verwendbarkeit der domänenspezifischen Sprache und sollte deshalb so gering wie möglich gehalten werden.

4 Generierung effizienter Optimierungsprobleme

Eine Optimierungsmöglichkeit bietet die Elimination von Entscheidungsvariablen für Komponenten. Im obigen Beispiel kann man die Entscheidungsvariable für P2, R3 und R4 zu einer zusammenfassen, da es sich dabei um einen einzigen Pfad handelt. Die Pfade können durch ähnliche Verfahren wie bei der Programmanalyse ermittelt werden. Alternativ können auch Pfade, welche einen anliegenden Druck nicht unterstützen, gänzlich weggelassen werden.

Ein weiterer wesentlicher Punkt ist die Linearisierung nichtlinearer Ausdrücke. In der Pumpendomäne gibt es nativ diverse nichtlineare Zusammenhänge, z. B. beim Druckabfall. Werden die entsprechenden Gleichungen direkt in das

```
# Komponenten des Systems als Elemente der entsprechenden Typmengen
set _PumpTypeA within pumps := { '_P1' };
set _SteelPipe within pipes := { '_R1', '_R2', '_R3', '_R4' };
set Source := { '_S' };
set Sink := { '_T' };

# Verbindungen des Systems als Tupelmenge
set connections within pipe_connectors cross pipe_connectors
  := { ('_S',  '_out', '_R1', '_in'),
       ('_R1', '_out', '_P1', '_in'), ...  };

# Lastfälle (Szenarien) des Systems
set scenarios := { '_S1', '_S2' };

# Druckerhaltungsgleichung für die Verbindungen
PressurePreservationConnections {S in scenarios, (U, P) in
    pipe_connectors, (U, P, V, Q) in connections}:
  _pressure_port[S, U, P] = _pressure_port[S, V, Q];

# Übernommene Gleichungen der Lastfälle
equation_S1_1: _pressure_port['_S1', '_S', '_out'] = 3;
equation_S1_2: _flow_port['_S1', '_S', '_out'] = 0.78;
  ...
# Abschalten nicht benötigter Entscheidungsvariablen
_S_isBought: _isBought['_S'] = 1;
_P1_isBought: _isBought['_P1'] = 1;
...
# Zielfunktion (aufsummierte Kosten für die gekauften Komponenten)
minimize objective: sum{C in components} _purchase[C] *
    _isBought[C];
```

Codeausschnitt 4. Auszug aus dem erstellten GMPL-Code für das im Codeausschnitt 3 vorgestellte System. Durch die vordefinierte Menge pumps können Gleichungen vereinfacht werden.

Optimierungsproblem übernommen, so ist auch das ganze Optimierungsproblem nichtlinear. Dieses kann dann nicht mehr so schnell und zuverlässig gelöst werden wie ein lineares Optimierungsproblem [2]. Daher verwenden wir Linearisierungstechniken wie Interpolation und Substitution, um die nichtlinearen Ausdrücke umzuformen.

Dazu zählen zunächst einfache Möglichkeiten des Übersetzerbaus wie die Konstantenfaltung oder das Ausmultiplizieren von Ausdrücken. Bleiben nichtlineare Terme zurück, muss man diese durch spezielle mathematische Verfahren annähern. Die einfachste Möglichkeit ist dabei, die Terme stückchenweise durch lineare Funktionen zu interpolieren. Position und Anzahl der Stützstellen haben entscheidenden Einfluss auf die Genauigkeit der Näherung. Aufgrund dieser Abrundungen muss das Ergebnis der Optimierung zum Schluss gegen die Ausgangsgleichungen getestet werden. So kann sichergestellt werden, dass das Ergebnis der Optimierung verwertbar ist.

5 Zusammenfassung und Ausblick

Dieser Artikel beschreibt eine domänenspezifische Sprache zur Planung optimaler Pumpensysteme. Entwickelt wurde die Sprache mit dem Übersetzerbautool Eli. Oft wiederkehrende Bauteile, wie Pumpen und Rohre, können schnell mit ihren jeweiligen Eigenschaften beschrieben werden. Die in ihnen geltenden Verhältnisse in Form von Gleichungen und Variablen sind bereits in den Sprachkonzepten verankert. Dadurch braucht man bei der Spezifikation einer neuen Pumpe nur die jeweiligen Anschlüsse und Pumpenkennlinien angeben und kann sie anschließend für den Systemaufbau verwenden. Dieser funktioniert ähnlich einer Programmiersprache, indem Komponenten deklariert werden können. Anschließend kann man diese miteinander verbinden, um ein Pumpensystem aufzubauen.

Dabei geschehen ständig Prüfungen durch die syntaktische und semantische Analyse. Darunter fallen Abfragen, welche Komponenten mit ihren Anschlüssen überhaupt verbunden werden können und welche Variablen innerhalb einer Komponente bekannt sind. Eine besondere semantische Analyse stellt hier die Vorabprüfung des zu erstellenden Optimierungsproblems dar. Hierbei können Methoden des Übersetzerbaus und der Programmanalyse verwendet werden. Dabei werden die Grenzen des Flüssigkeitsflusses der Anlage iterativ ausgelotet. Die Analyse bringt den Vorteil, dass für den Nutzer lesbare Fehlermeldungen erzeugt werden können. Würde das Optimierungsproblem direkt von einem LP-Solver bearbeitet werden, dann würde man für nicht lösbare Instanzen kaum Informationen erhalten und wenn dann nur über generierte Variable und Gleichungen.

Bei den auftretenden Linearisierungen für die Kennlinien und nichtlinearen Ausdrücke entstehen Rundungsfehler. Daher wird zusätzlich zum Optimierungsmodell auch ein Simulationsmodell erstellt, welches das modellierte System überprüfen soll. In der Simulation werden dann die optimierten Parameter eingestellt und die auftretenden Drücke und Durchflüsse getestet. Sollten hier zu große Abweichungen auftreten, muss die Optimierung mit genaueren Linearisierungen wiederholt werden.

Die Sprache wird demnächst um neue Eingabemethoden von Kennlinien erweitert. Diese können dann z. B. aus CSV-Dateien geladen werden, was zu einer erhöhten Übersicht der zu schreibenden Dateien führt. Auch ist es vorstellbar, die Kennlinien direkt als Funktion anzugeben und Stützstellen selbst zu bestimmen. Dadurch lassen sich dann auch theoretische oder noch konstruierte Pumpen ohne konkrete Messwerte simulieren und analysieren.

In weiteren Schritten sollen dann die Pumpen detaillierter modelliert werden. Diese bestehen dann aus mehreren Bauteilen, wie Frequenzumrichter und Motor, welche zusätzliche Einstellungsparameter ermöglichen. Dadurch ist eine weiterführende Optimierung möglich. Auch andere Bauteile erhalten zusätzliche Parameter, wie z. B. die Beschaffenheit des Rohres, welche Einfluss auf den darin geltenden Druckabfall nimmt. Vorgefertigte Komponenten, wie z. B. spezielle Pumpen, werden durch Bibliotheken modularisiert zugreifbar gemacht.

Literatur

[1] Johan Åkesson u. a. „Modeling and optimization with Optimica and JModelica. Languages and tools for solving large-scale dynamic optimization problems“. In: *Computers & Chemical Engineering* 34.11 (2010), S. 1737–1749.

[2] Pietro Belotti u. a. „Mixed-integer nonlinear optimization“. In: *Acta Numerica* 22 (2013), S. 1–131.

[3] Christian Berg und Wolf Zimmermann. „Evaluierung von Möglichkeiten zur Implementierung von Semantischen Analysen für Domänenspezifische Sprachen.“ In: *Software Engineering (Workshops)*. 2014, S. 111–128.

[4] Deutsche Energie-Agentur GmbH (dena). „Energiesparpaket: Leuchtturmprojekt zur energetischen Optimierung von Pumpensystemen“. In: (2011).

[5] Robert Fourer, David Gay und Brian Kernighan. *Ampl*. Bd. 119. Boyd & Fraser, 1993.

[6] S Götz u. a. „Modeldriven self-optimization using integer linear programming and pseudoboolean optimization“. In: *Proceedings of ADAPTIVE* (2013), S. 55–64.

[7] Gabor Karsai u. a. „Design guidelines for domain specific languages“. In: *arXiv preprint arXiv:1409.2378* (2014).

[8] Uwe Kastens, Peter Pfahler und Matthias Jung. „The eli system“. In: *Compiler Construction*. Springer. 1998, S. 294–297.

[9] Andrew Makhorin. *GLPK - GNU Project - Free Software Foundation (FSF)*. 2015. URL: `http://www.gnu.org/software/glpk/`.

[10] Frederic H Murphy und Edward A Stohr. „An intelligent system for formulating linear programs“. In: *Decision Support Systems* 2.1 (1986), S. 39–47.

[11] Peter Pelz u. a. *Designing Pump Systems by Discrete Mathematical Topology Optimization: The Artificial Fluid Systems Designer*. Techn. Ber. International Rotating Equipment Conference, 2012.

[12] Bundesministerium für Wirtschaft und Energie. *Gesamtausgabe der Grafiken zu Energiedaten*. 26.08.2015. URL: `http://www.bmwi.de/BMWi/Redaktion/PDF/E/energiestatistiken-grafiken,property=pdf,bereich=bmwi2012,sprache=de,rwb=true.pdf`.

Danksagung

Dieses Projekt wird gefördert durch das Bundesministerium für Wirtschaft und Technologie unter dem Förderkennzeichen 03ET1134C. Partner in diesem Projekt sind die Technische Universität Darmstadt und die KSB Aktiengesellschaft. Dank gilt außerdem Felix Knispel für die Zuarbeit zum Projekt und zu diesem Artikel.